Report No. 75-0004
Contract No. NAS8-25621

(NASA-CR-120703) VORTEX INFORMATION DISPLAY SYSTEM PROGRAM DESCRIPTION MANUAL (M&S Computing, Inc.) 249 p HC $7.50 CSCL 09B

N75-20013

Unclas
G3/61 14583

VORTEX INFORMATION DISPLAY SYSTEM PROGRAM DESCRIPTION MANUAL

February 11, 1975

M&S COMPUTING, INC.

PREFACE

The Vortex Information Display System (VIDS) Program Description Manual explains the design of VIDS software that is used to collect and process wing-tip-trailing vortex data received from Laser Doppler Velocimeter Systems. VIDS was developed for NASA at Marshall Space Flight Center (Contract No. NAS8-25621, Mods. 17 and 18), under a joint NASA/FAA venture to study the effects of air disturbances created by moving aircraft.

Prepared by:

R. Conway
G. N. Matuck
J. M. Roe
J. Taylor
A. Turner

Project Manager:

E. I. Eastin

Approved by:

J. W. Meadlock

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

TABLE OF CONTENTS
(continued)

LIST OF FIGURES

LIST OF FIGURES
(continued)

LIST OF FIGURES
(continued)

LIST OF TABLES

PRECEDING PAGE BLANK NOT FILMED

1. INTRODUCTION

The Vortex Information Display System (VIDS) provides flexible control through system-user interaction for collecting wing-tip-trailing vortex data, processing this data in real time, displaying the processed data, storing raw data on magnetic tape, and post processing raw data. The data is received from two asynchronous Laser Doppler Velocimeters (LDV's) and includes position, velocity, and intensity information. The raw data is written onto magnetic tape for permanent storage and is also processed in real time to locate vortices and plot their positions as a function of time.

The interactive capability enables the user to make real time adjustments in processing data and thereby provides a better definition of vortex behavior. Displaying the vortex information in real time produces a feedback capability to the LDV system operator allowing adjustments to be made in the collection of raw data. Therefore, both raw data and processing can be continually upgraded during flyby testing to improve vortex behavior studies. The post-analysis capability permits the analyst to perform in-depth studies of test data and modify vortex behavior models to improve transport predictions.

VIDS is composed of both PDP-11 support software and M&S Computing application software running under control of the PDP-11 operating system.

1.1 PDP-11 Support Software

The PDP-11 support software includes system programs and utilities designed by Digital Equipment Corporation (DEC) to support the PDP-11 user during execution of application programs. Specifically, the PDP-11 software components are:

- Disk Operating System (DOS)
- Verification Program (VERIFY)

Sections 2 and 4 of the Vortex Information Display System User's Manual discuss these PDP-11 components in detail.

1.2 M&S Computing Application Software

The M&S Computing application software is composed of routines that control program flow and that collect, store, process, and display data. These routines are grouped as follows:

- Data Acquisition routines which input and output raw data.
- FORTRAN routines which initialize variables, locate vortex centers, and provide interfaces between the assembly language I/O routines, the vortex location routines, and the display controller.

1.3 M&S Computing Support Software

The M&S Computing support software is composed of the display librarian which creates predefined displays in a library to be used and displayed by the application software.

1.4 Hardware Configuration

The hardware configuration (see Figure 1-1) required by VIDS is:

- o Digital Computer, PDP-11 series model 35 with 32K memory and 16 bit words - DEC.
- o Magnetic Disk Unit, RK05 - DEC.
- o Magnetic Tape Unit, TU10 - DEC.
- o Graphics Display Terminal, 4014 series terminal and 613 monitor - Tektronix, Inc.
- o Hard Copy Device, 4610 series - Tektronix, Inc.
- o Graphics Data Tablet, HW-1-11, Summagraphics.

Operation of VIDS hardware is discussed in Section 2, Vortex Information Display System User's Manual.

VIDS HARDWARE SYSTEM COMPONENTS

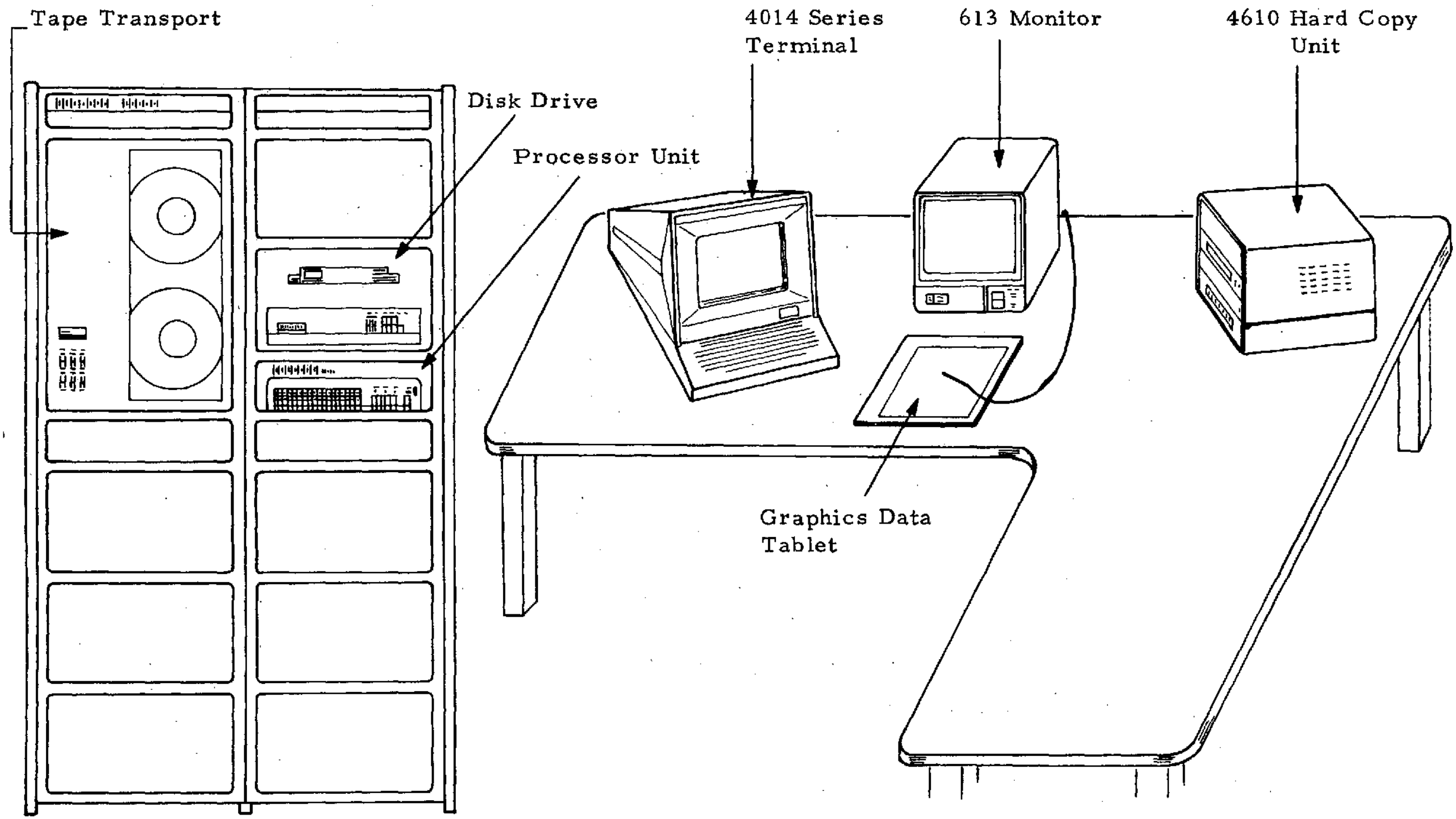


Figure 1-1

2. DATA ACQUISITION

The Data Acquisition portion of the Banning Vortex software system is written in assembly language to handle interrupts as quickly as possible. It is divided into three major portions:

1. ATVWS - initialization.
2. READ - acquire the data from the LDV's and write the data on the disk.
3. FILL - read the data from the disk and format it as required by the data processing functions.

Figure 2-1 depicts the overall data flow for the Data Acquisition function and illustrates the role of each software portion. Each of the three Data Acquisition portions will be described in this section, and the section will be organized as follows:

- Overview of function
- Interaction with other functions
- Critical parameters
- Parameters which define function capability
- Module descriptions, which will include:
  - Purpose of module
  - Calling sequence
  - Other modules required
  - Critical parameters

Listings of the Data Acquisition software are contained in the Appendices.

If any of the specific items are not required for a particular portion of module description, then that heading and item will be omitted.

2.1 ATVWS Overview

This function is responsible for initialization of the hardware and software components which make up the Data Acquisition portion of the Banning Vortex system.

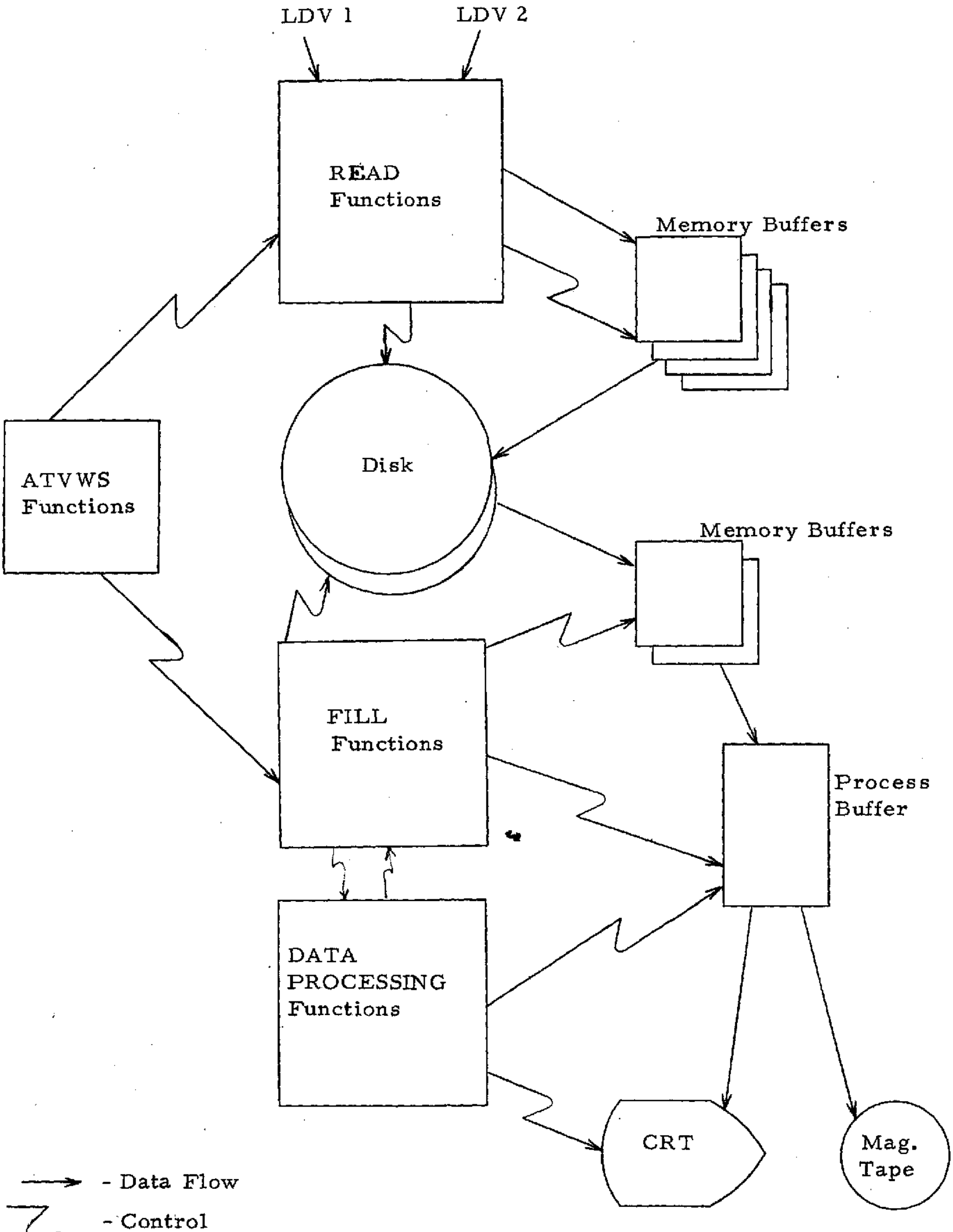

LDV 1
LDV 2
READ Functions
Memory Buffers
Disk
ATVWS Functions
Memory Buffers
FILL Functions
Process Buffer
DATA PROCESSING Functions
CRT
Mag. Tape
- Data Flow
- Control

Figure 2-1

In particular, modules within this function perform the following:

- o Initialize the hardware interfaces (DR-11's).
- o Start flyby or test.
- o Stop flyby or test.
- o Start data collection.
- o Stop data collection.
- o Read plane type.
- o Initialize pointers and flags required by the software modules.

2.1.1 Interaction

This function interacts with the other two functions (READ and FILL) in that:

- o ATVWS initializes flags and indicators for those functions.
- o ATVWS contains the central logic to display all error messages which describe error conditions detected by any of the three Data Acquisition functions.

2.1.2 Critical Parameters

ATVWS relies on the flag IFLD which is common to Data Acquisition and Data Processing functions, and which indicates the LDV's required for any particular test. This flag controls routine INIT and must be set prior to entry, as follows:

IFLD	=	0, both LDV's
IFLD	=	1, LDV 2 only
IFLD	=	2, LDV 1 only

2.1.3 Design Parameters

Additional error messages may easily be added simply by entering the address of the message in the list MSGAD. A particular message is displayed b executing a Jump (JMP) instruction to location HALT (which is GLOBAL) with the appropriate message index in R0. (Each index is a multiple of 2 which provides for word indexing.)

2.1.4 Module Descriptions

Module ATVWS

Purpose

The ATVWS module (Figure 2-2) initializes the interrupt vectors for the three DR-11's. It also initializes the disk file LDVS which is allocated to store the accumulated data. ATVWS is called by the Display Controller at the start of the system execution.

Calling Sequence

```
JSR     R5, ATVWS
```

(cannot be called via FORTRAN routine)

Modules Required

LKTRAN

INITDS

DISPIO

Module INIT

Purpose

The INIT module (Figure 2-3) initializes all software pointers and flags and enables the DR-11 interrupts. The latter permits detection of the request A and request B interrupts.

Calling Sequence

```
    JSR     R5, INIT

or  CALL INIT
```

Modules Required

INITBF

Critical Parameters

IFLDV is a control indicator which must be set prior to INIT call:

IFLD = 0, both LDV's

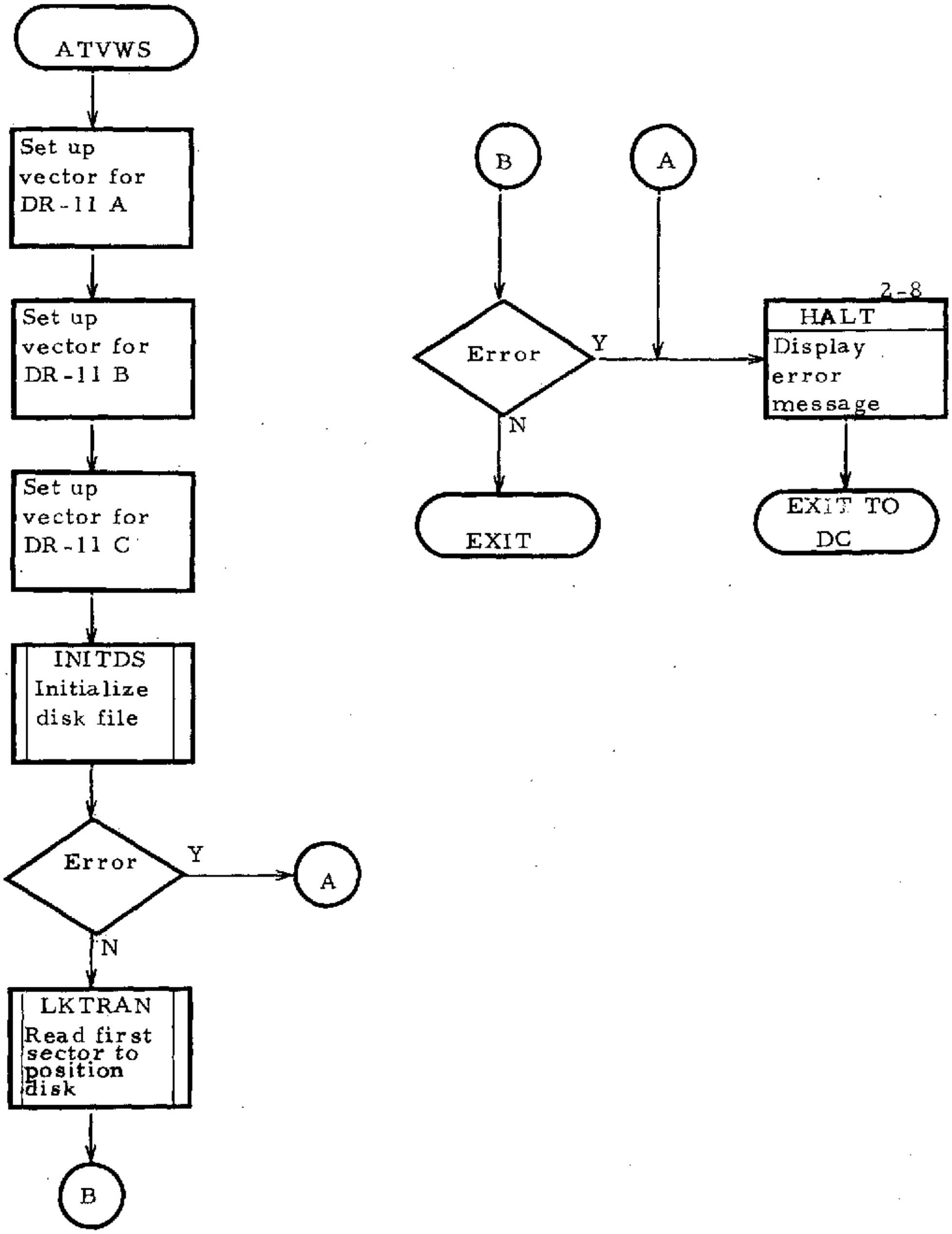

ATVWS
Set up vector for DR-11 A
Set up vector for DR-11 B
Set up vector for DR-11 C
INITDS
Initialize disk file
Error
Y
N
A
LKTRAN
Read first sector to position disk
B
B
A
Error
Y
N
2-8
HALT
Display error message
EXIT
EXIT TO DC

Figure 2-2

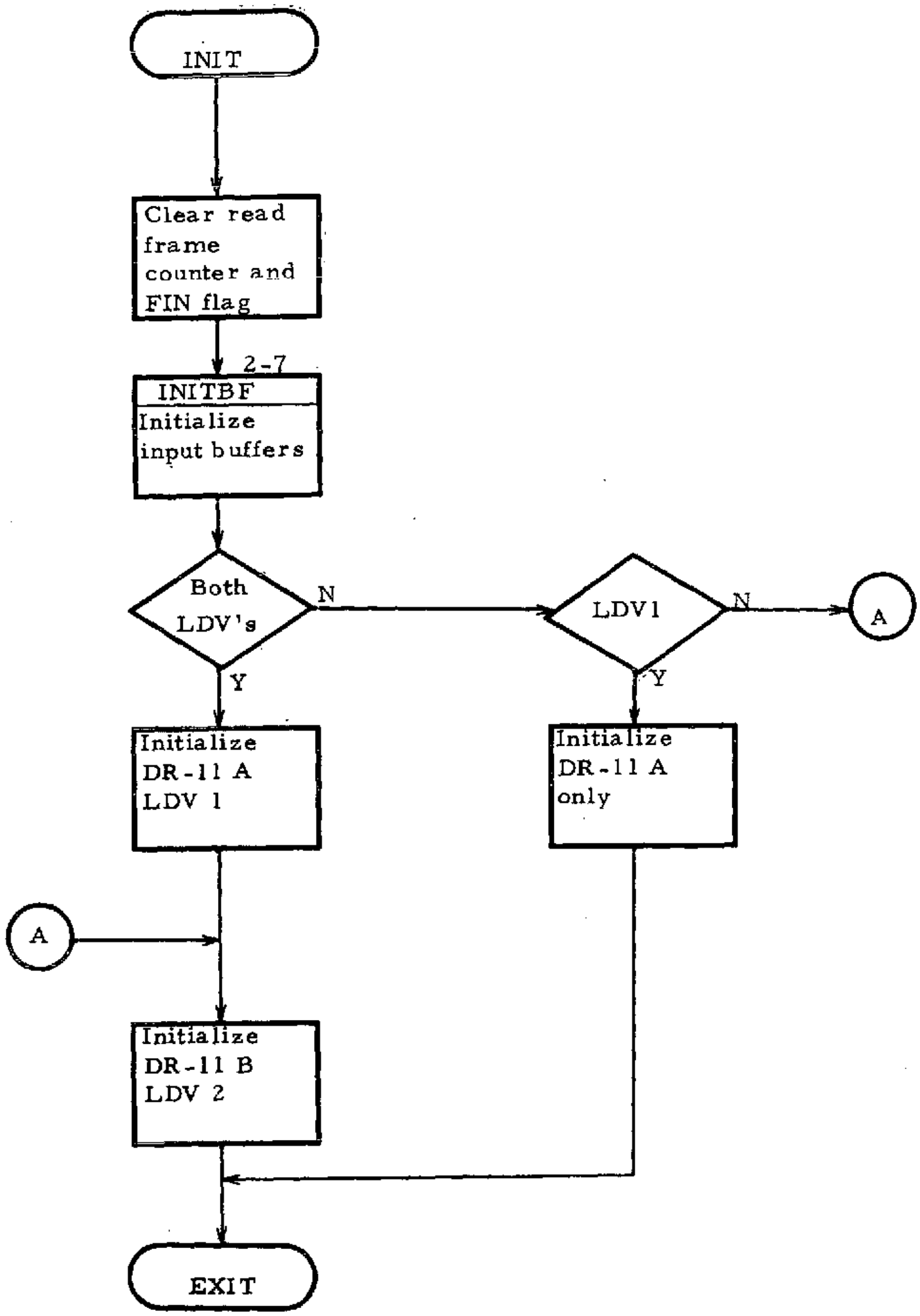

INIT
Clear read frame counter and FIN flag
2-7
INITBF
Initialize input buffers
Both LDV's
N
Y
LDV1
N
Y
A
Initialize DR-11 A LDV 1
Initialize DR-11 A only
A
Initialize DR-11 B LDV 2
EXIT

Figure 2-3

IFLD = 1, LDV 2 only

IFLD = 2, LDV 1 only

IFLD is the first word in common IFLDV.

Module DISABL

Purpose

The DISABL module (Figure 2-4) disarms the hardware interrupts for the two DR-11's. A software flag FIN (normally zero) is set to a positive one to denote end of test.

Calling Sequence

```
        JSR     R5,DISABL
or      CALL DISABL
```

Module DISAB1

Purpose

The DISAB1 module (Figure 2-5) disarms the two interrupts for the DR-11 used to input the Start-of-Flyby signal and the plane type.

Calling Sequence

```
        JSR     R5,DISAB1
or      CALL DISAB1
```

Module INTCA

Purpose

The INTCA module (Figure 2-6) responds to the interrupt generated by the DR-11 that specifies the Start-of-Flyby and plane identification. A Start-of-Flyby signal ends any current test and initializes the hardware and software in preparation for the start of another flyby.

Modules Required

DISABL

DISABLE
INTERRUPT 1

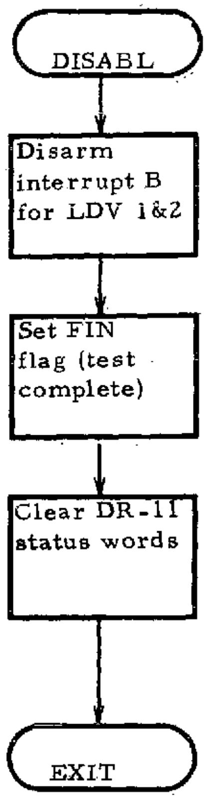


Figure 2-4

DISABLE
INTERRUPT 2

Figure 2-5

ENABLE FLYBY
INTERRUPT

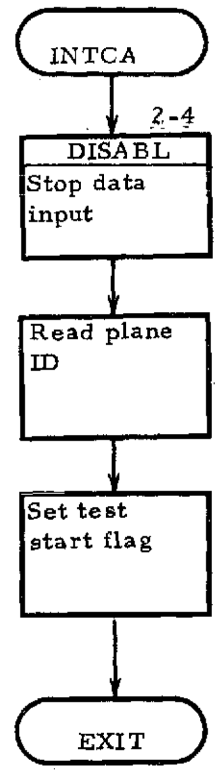


Figure 2-6

Module INITBF

Purpose

The INITBF module (Figure 2-7) initializes all the data input buffers and assigns a buffer to each of the LDV's. This module also initializes all software pointers and flags.

Calling Sequence

```
JSR     PC,INITBF
```

(Module cannot be called via FORTRAN routine).

Module HALT

Purpose

The purpose of the HALT module (Figure 2-8) is to stop or abort the current flyby, to display an appropriate message on the CRT, and to initialize the system for further test or flybys. This module is not a subroutine; it is the central error handling function.

Calling Sequence

```
MOV     ERR,R0   (set error message index (see Section 2.1.3)).

JMP     HALT     (jump to error routine).
```

where:

ERR is a unique error number. Error numbers 2-8 are currently used and are assigned such that R0 may be used as a word index. That is, R0 contains an integer which is a multiple of 2.

Modules Required

DISPIO

2.2 READ Overview

This function is responsible for acquiring the data for the Banning Vortex software system.

Modules within this function respond to the DR-11 interrupts, input the data words in response to the interrupts, accumulate the data in memory buffers, and write the data on the disk as the memory buffers are filled.

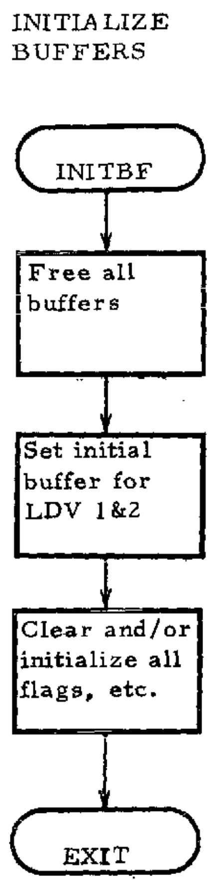


Figure 2-7

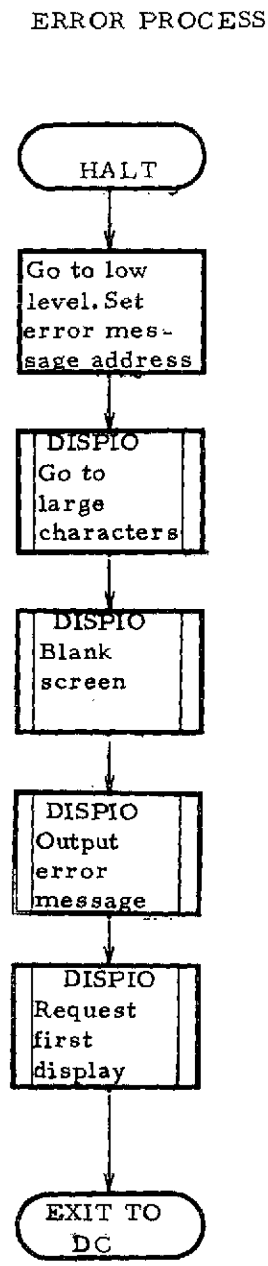


Figure 2-8

2.2.1 Interaction

This function interacts with ATVWS in that ATVWS initializes READ pointers and flags and enables the DR-11 interrupts which cause READ function responses. READ interacts with FILL in that FILL functions process data on the disk written by the READ functions. READ functions also maintain a count for the FILL functions to enable them to track the progress of each LDV

2.2.2 Critical Parameters

Modules within the READ function manage a pool of data input buffers. These buffers each contain 256 words, which is the physical size of a disk sector. Figure 2-9 illustrates the format of these data buffers. As can be seen by this figure, each buffer contains 248 data words and 8 control words. The number of buffers assigned to this buffer pool is critical to the system. Enough buffers to support input functions are required when the rate of input is at a maximum for both LDV's. This rate of input must be sustained during those periods (such as data being FILLed) when data cannot be written on the disk.

2.2.3 Design Parameters

Additional buffers may be added to the system; modules within READ are designed to manage any number. Location BUF contains the current number of buffers and must be changed as buffers are added or removed from the system.

Each buffer consists of two Reserve Block Word operators (.BLKW):

```
.BLKW        SIZB
.BLKW        SIZD
```

SIZB (number of control words) has been equated to 8; SIZD (number of data words) to 248. Changing either or both of these parameters will in turn change all data buffers.

2.2.4 Module Descriptions

Module INTAA

Purpose

The INTAA module (Figures 2-10 through 2-12) responds to the request A (end of frame (EOF)) interrupt for DR-11 A (LDV 1). Upon entry, this module inputs four data words (to clear the interrupt signal) and discards the data. If the interrupt is the first EOF for the flyby, the request B interrupt (data input) is enabled. Otherwise, the current buffer is marked to denote an EOF has occurred,

DATA BUFFER FORMAT

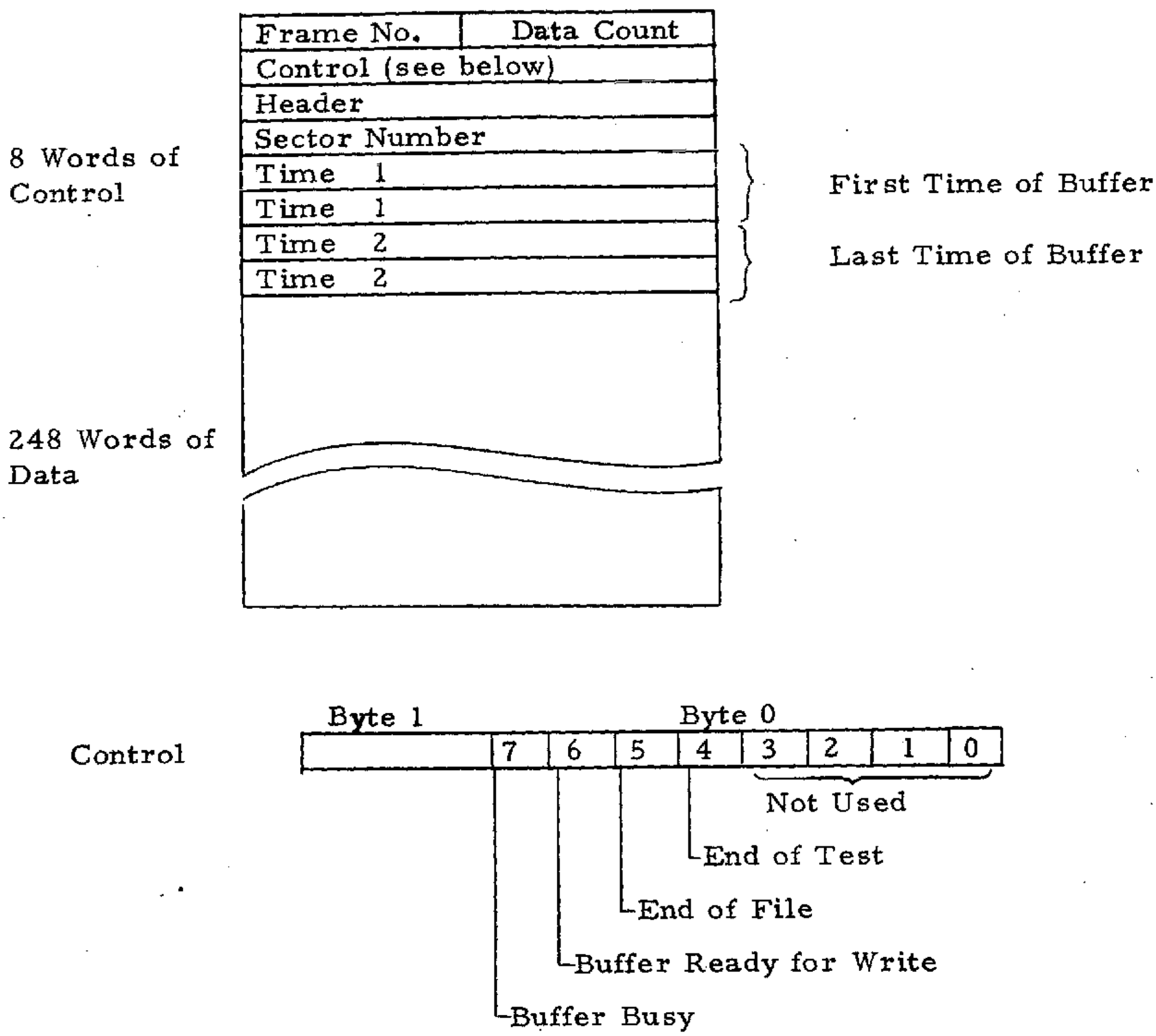


Figure 2-9

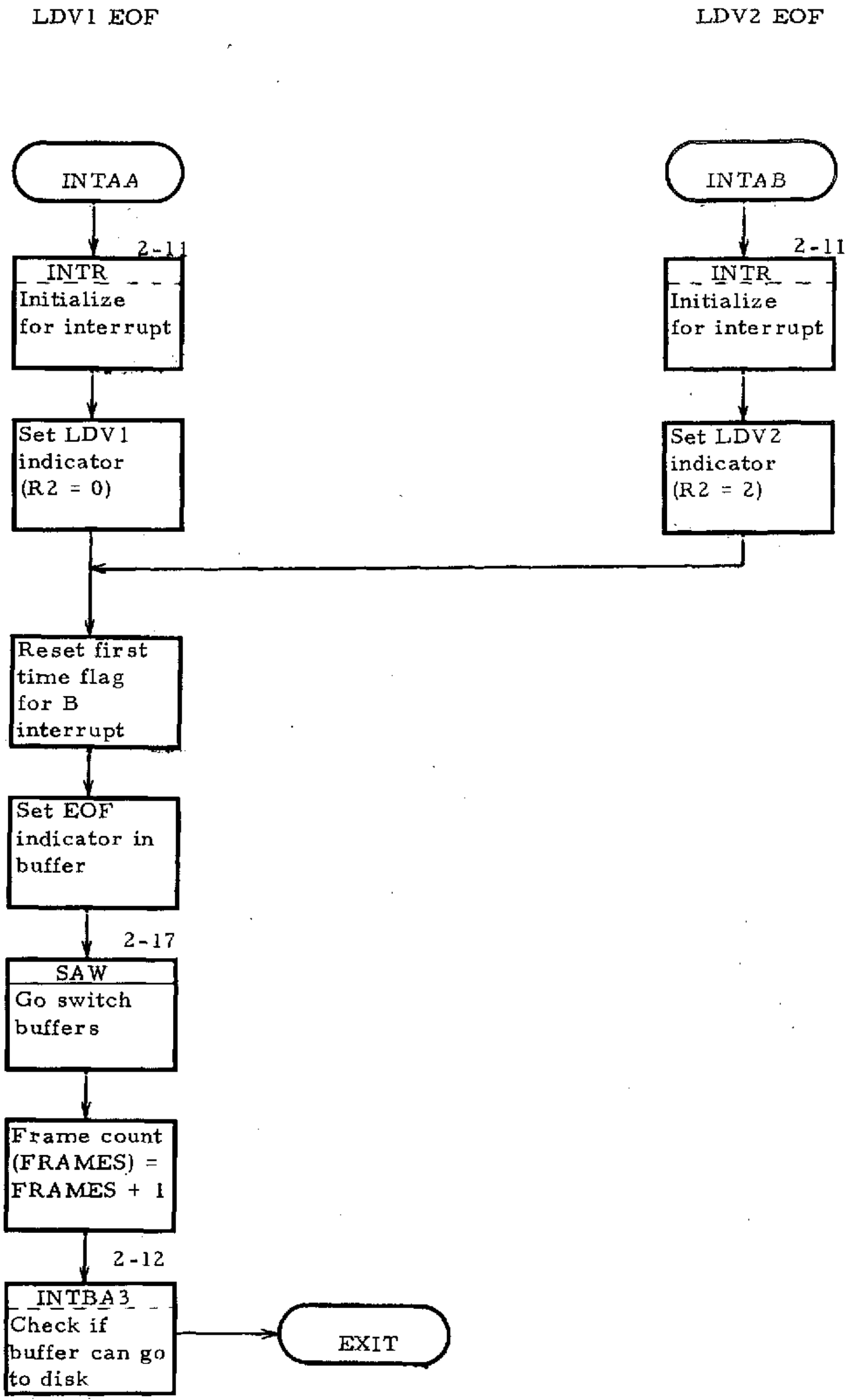

LDV1 EOF
LDV2 EOF
INTAA
INTAB
2-11
INTR
Initialize for interrupt
2-11
INTR
Initialize for interrupt
Set LDV1 indicator (R2 = 0)
Set LDV2 indicator (R2 = 2)
Reset first time flag for B interrupt
Set EOF indicator in buffer
2-17
SAW
Go switch buffers
Frame count (FRAMES) = FRAMES + 1
2-12
INTBA3
Check if buffer can go to disk
EXIT

Figure 2-10

PROCESS DATA

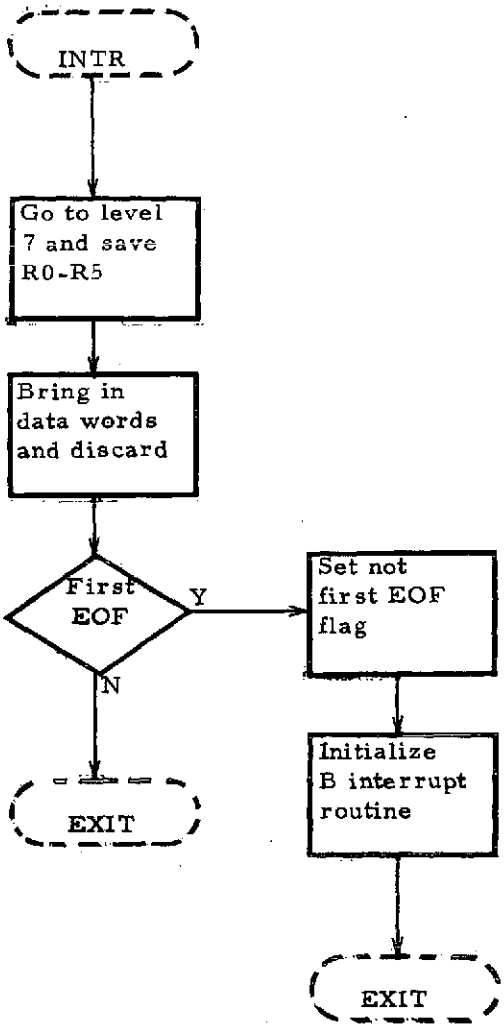


Figure 2-11

DISK WRITE PREAMBLE

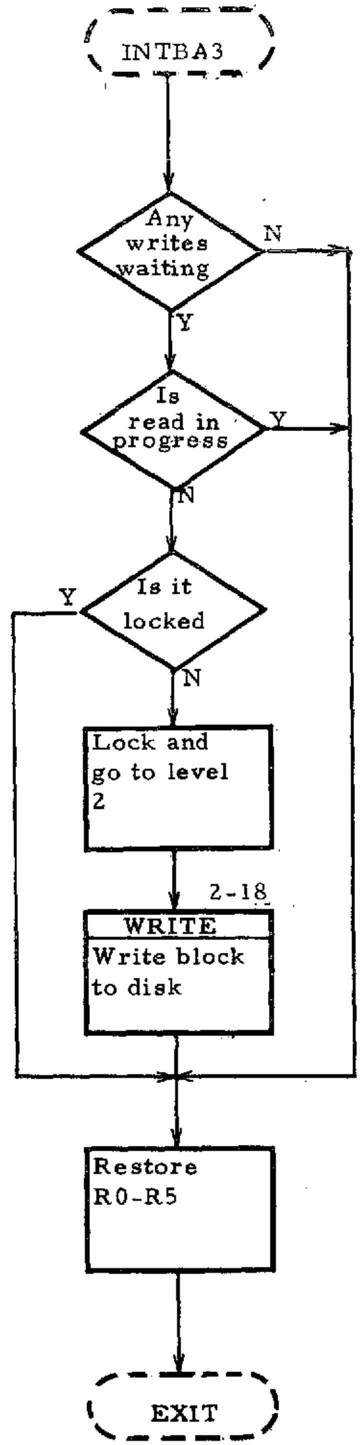


Figure 2-12

a new buffer is assigned to the LDV, and the full buffer is written on disk, if the disk write services are available. If the write services are not available, then the buffer is marked for a "write ready," and the interrupt module is exited. The buffer will be written at the next available time.

Modules Required

SAW

WRITE

Module INTAB

Purpose

The INTAB module (Figures 2-10 through 2-12) performs the same function as INTAA (see X. 5. 1 above) except INTAB provides support for DR-11 B. Certain functions (see flowchart) are common to both INTAA and INTAB.

Modules Required

SAW

WRITE

Module INTBA

Purpose

The INTBA module (Figures 2-12 and 2-13) inputs the application data via DR-11 A (LDV 1). This module responds to the request B interrupt, brings in the 4 data words, sets the time and increments the word count. If the data buffer is full, the buffer is marked "write read," a new buffer is assigned to the LDV, and the full buffer is written on disk, if the disk services are available. If the disk services are not available, the buffer is written at the next available time. If the buffer is not full, this module is exited after bringing in the data. This module provides code that is common to INTAA, INTAB, and INTBA.

Modules Required

SAW

WRITE

Module INTBB

Purpose

The INTBB module (Figures 2-12 and 2-13) performs the same function as INTBA except INTBB provides support for DR-11 B (LDV 2). Certain functions are

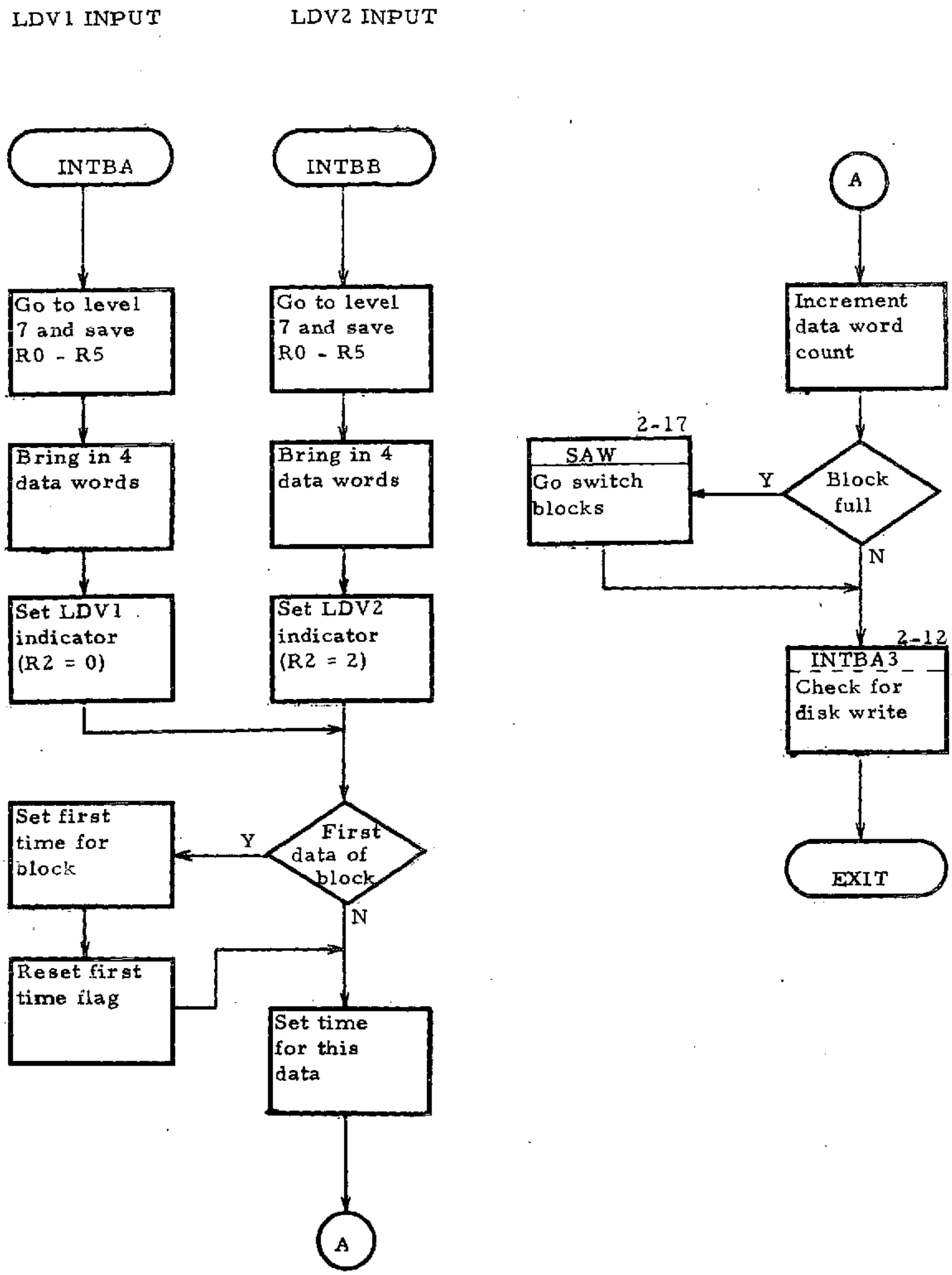

LDV1 INPUT
LDV2 INPUT
INTBA
INTBB
Go to level 7 and save R0 - R5
Go to level 7 and save R0 - R5
Bring in 4 data words
Bring in 4 data words
Set LDV1 indicator (R2 = 0)
Set LDV2 indicator (R2 = 2)
Set first time for block
Y
First data of block
N
Reset first time flag
Set time for this data
A
A
Increment data word count
2-17
SAW
Go switch blocks
Y
Block full
N
2-12
INTBA3
Check for disk write
EXIT

Figure 2-13

common (see flowchart) and the code is therefore shared.

Modules Required

SAW

WRITE

Module INTAW

Purpose

The INTAW module (Figure 2-14) responds to the request B interrupt for DR-11 A (LDV 1). This module processes the data prior to receiving the start of flyby (interrupt request A or EOF). The data is read in and discarded.

Module INTBW

Purpose

The INTBW module (Figure 2-15) responds to the request B interrupt for DR-11 A (LDV 2). This module processes the data prior to receiving the start of flyby interrupt. The data is read in and discarded.

Module CALBLK

Purpose

The CALBLK module (Figure 2-16) calculates the sector address for each buffer for both LDV's. A current sector address is maintained for LDV 1 and LDV 2, where LDV 1 buffers are written on even sectors and LDV 2 buffers are written on odd sectors.

Calling Sequence

```
JSR     PC, CALBLK
```

(cannot be called by FORTRAN routine).

Upon return, the sector number is in R4.

Module SAW

Purpose

The SAW module (Figure 2-17) manages the buffer switching function. When called, this module assigns a new buffer to the LDV and processes the control word for the full buffer, as follows:

LDV1 WAIT

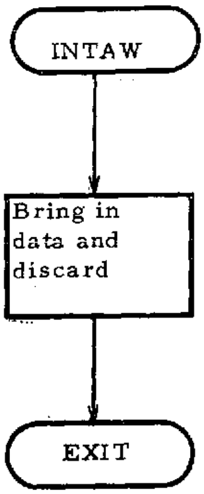


Figure 2-14

LDV2 WAIT

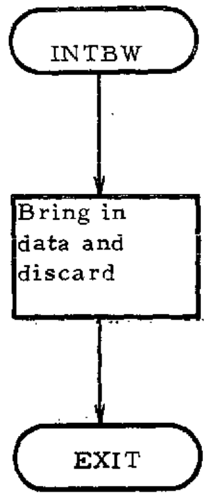


Figure 2-15

CALCULATE DISK SECTOR

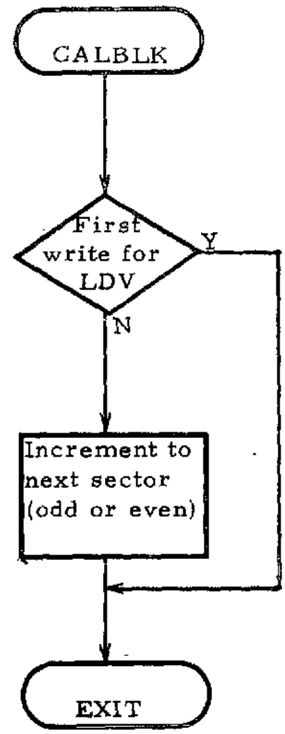


Figure 2-16

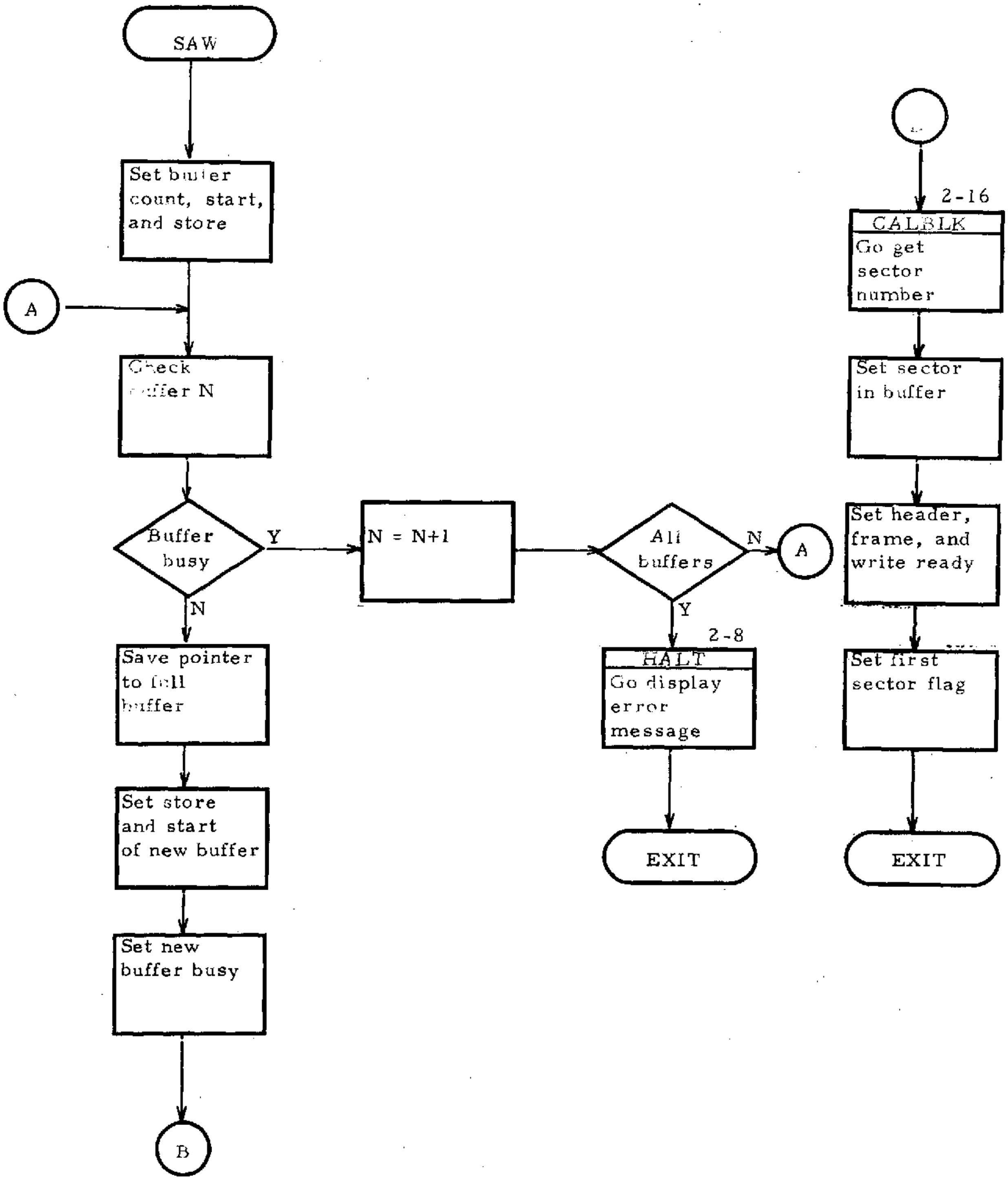


Figure 2-17

ORIGINAL PAGE IS OF POOR QUALITY

- o Sets store address and start address for new buffer. The store address is used to place data into the buffer; the start address is used to store buffer control information.
- o Sets new buffer busy (see Figure 2-9).
- o Calls CALBLK to obtain sector address and places the address in the old buffer control word.
- o Sets header and frame number in the old buffer control words. Note that the frame number is positive for LDV 1 and negative for LDV 2.
- o Sets the write ready status (see Figure 2-9) for the old buffer.

Calling Sequence

```
JSR     PC, SAW
```

(cannot be called via FORTRAN routine).

Upon entry: R2 = 0, for LDV 1

R2 = 2, for LDV 2

Modules Required

CALBLK

HALT

Critical Parameters

This module executes on level 7.

Module WRITE

Purpose

The WRITE module (Figure 2-18) performs all functions required to write the data buffers on the disk.

Upon entry, the module searches the buffer pool to find the buffers waiting ("write ready") to be written on the disk. From the buffers ready, this module finds the next logical buffer. The next logical buffer must:

- o Be in same sequence as the last buffer written. That is, if the last buffer written was for LDV 1 (even sector number), then all LDV 1 ready buffers will be written before any LDV 2 buffers (odd sector number). Likewise, if the last sector written was

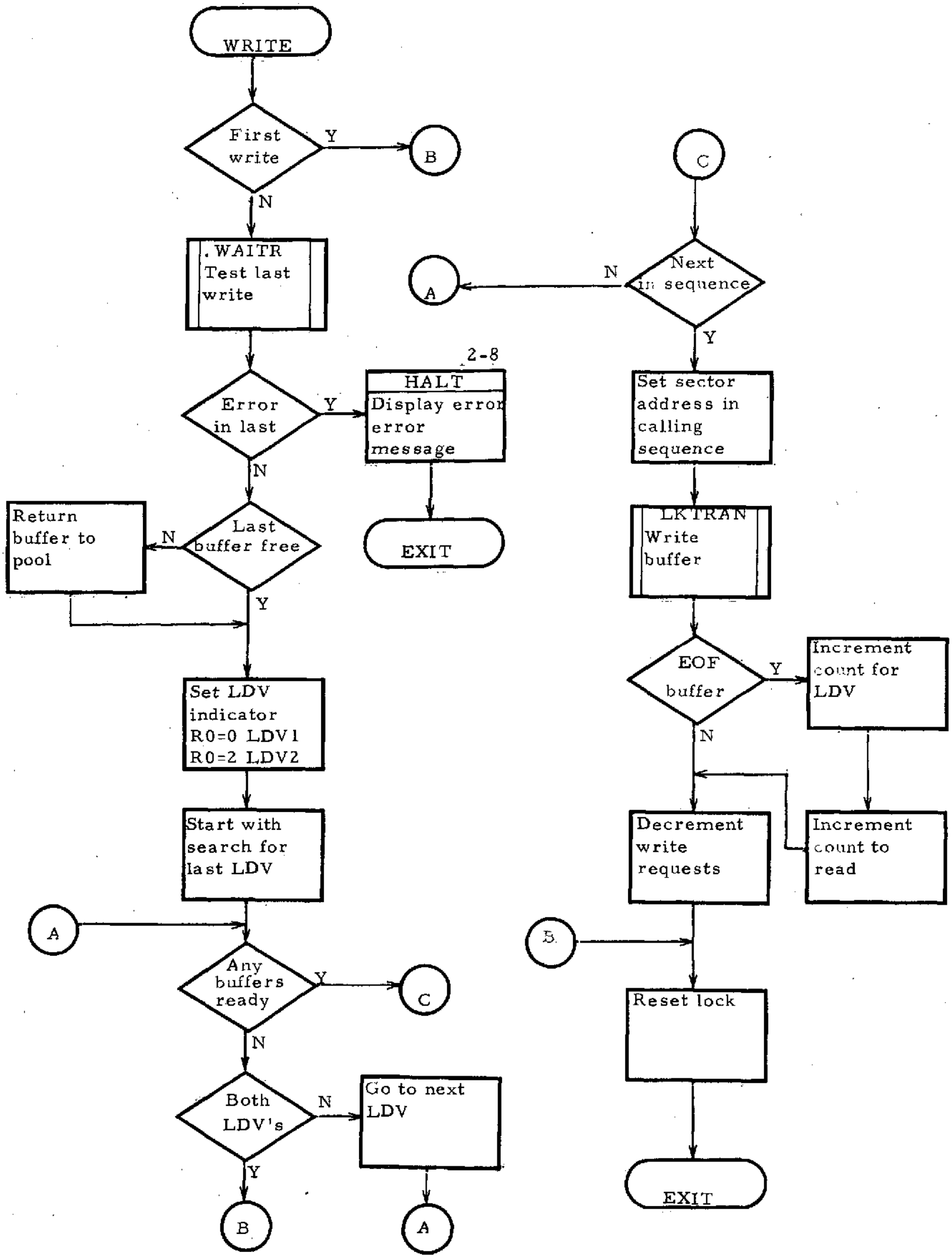

WRITE
First write
Y
B
N
.WAITR Test last write
2-8
HALT
Display error error message
Error in last
Y
N
EXIT
Return buffer to pool
N
Last buffer free
Y
Set LDV indicator R0=0 LDV1 R0=2 LDV2
Start with search for last LDV
A
Any buffers ready
Y
C
N
Both LDV's
N
Go to next LDV
Y
B
A
C
A
N
Next in sequence
Y
Set sector address in calling sequence
LKTRAN Write buffer
EOF buffer
Y
Increment count for LDV
N
Decrement write requests
Increment count to read
B
Reset lock
EXIT

Figure 2-18

for LDV 2, then all odd sectors will be written before any even sectors.

- Be in sequence for either even or odd sector numbers. Sectors are written in the order specified by their sector numbers, which progress in increments of 2; i.e., 0, 2, 4, ... for LDV 1 or 1, 3, 5, ... for LDV 2. This method of selection is designed to minimize the disk access times since once the requests are synchronized (after the first sector is written), all other sectors can be written within two sector times, or approximately 6 milliseconds.

After a buffer is written on the disk, its control word is examined to determine if an EOF occurred for the buffer. If an EOF occurred, the number of frames written on the disk is incremented. The buffer is marked as "free" and returned to the buffer pool after it is written on the disk.

Calling Sequence

```
JSR    PC,WRITE
```

(cannot be called via FORTRAN routine).

Upon entry: R2 = 0, for LDV 1

R2 = 2, for LDV 2

Modules Required

WAITR

LKTRAN

HALT

Critical Parameters

This module must be serially executed; it is not reentrant. To insure this, the module is protected by a flag (LOCK) which is set upon entry and reset upon exit. In addition, this module may not be executed while a read disk operation is in progress. BUSY is set by the read disk function and is reset when the read completes. This module is not executed if BUSY is set.

2.3 FILL Overview

This function is responsible for reading the data from disk and placing the data in the process buffer in the specified format. Figure 2-19 provides a

FILL PROCESS

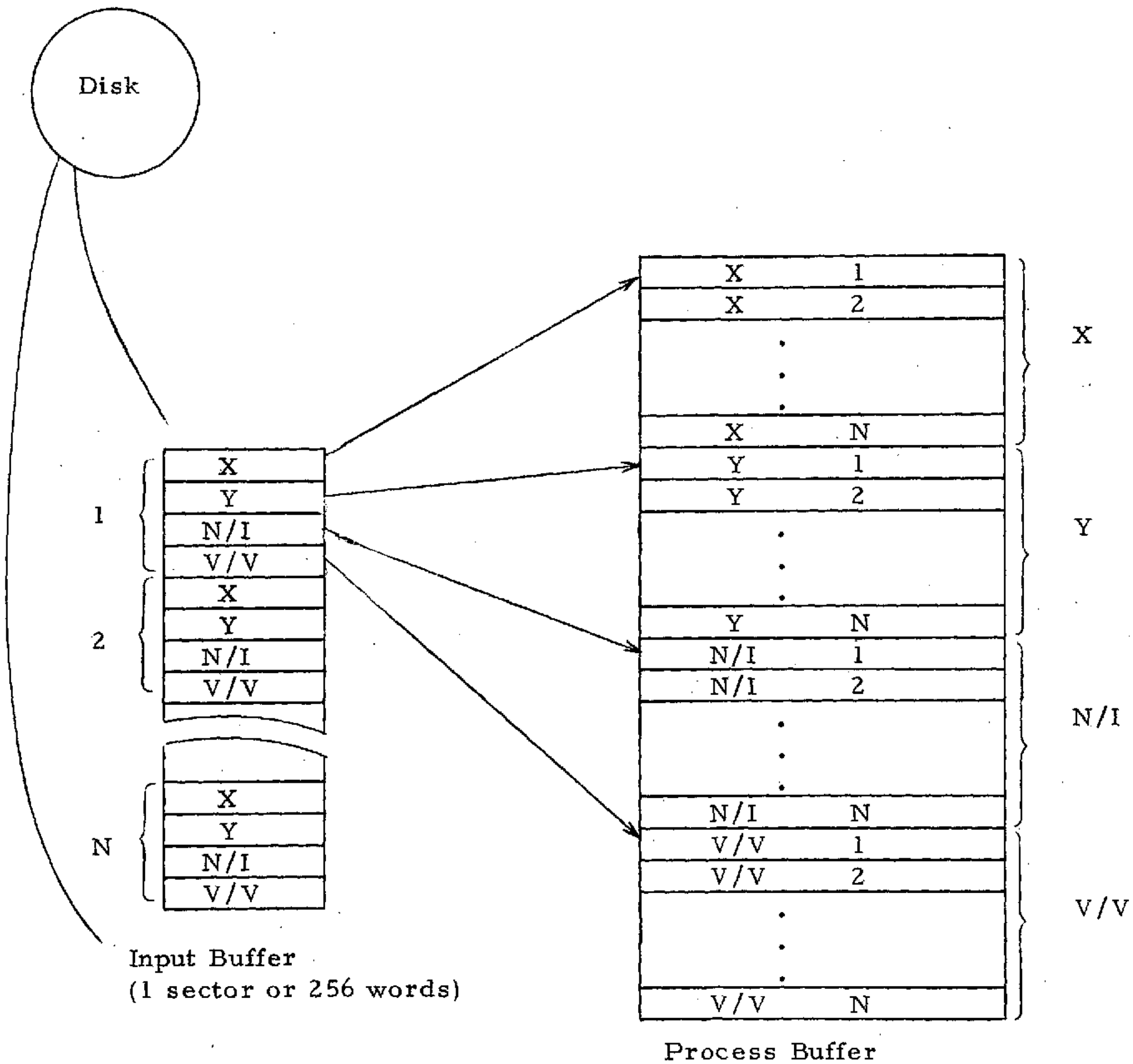


Figure 2-19

pictorial description of this process. From this figure it can be seen that:

- o data is brought from disk in data buffers which are one sector in length; and
- o the data read from disk is stripped and placed in the process buffer in such a manner that like data is contiguous.

One fill request results in either:

- o all data for one frame being placed in the process buffer,
- o a maximum of 992 data points being placed in the buffer, or
- o no data being placed in the process buffer because an abort was detected.

2.3.1 Interaction

This function interacts with READ in that a count is maintained for frames written on the disk for each LDV. Basically, READ increments the count as data is written and FILL decrements the count as data is read.

2.3.2 Critical Parameters

This function is given priority over the disk write functions. BUSY is a flag (set = -1) which inhibits the write process. That is, a disk write function will not be processed if a read is in progress. This enables a FILL iteration to be processed with a minimum of disk accesses.

2.3.3 Module Descriptions

Module FILL

Purpose

The FILL module (Figures 2-20 through 2-23) provides the FORTRAN data processing functions with test data read from disk. The data is read from disk and placed in the process buffer in the format required by the calling functions.

Data read in from disk is double-buffered such that one buffer is being processed while the other buffer is being filled by the disk. This technique enables both the data read and process functions to be accomplished in essentially the time required for the disk read.

Calling Sequence

```
        JSR    R5,FILL

or      CALL FILL
```

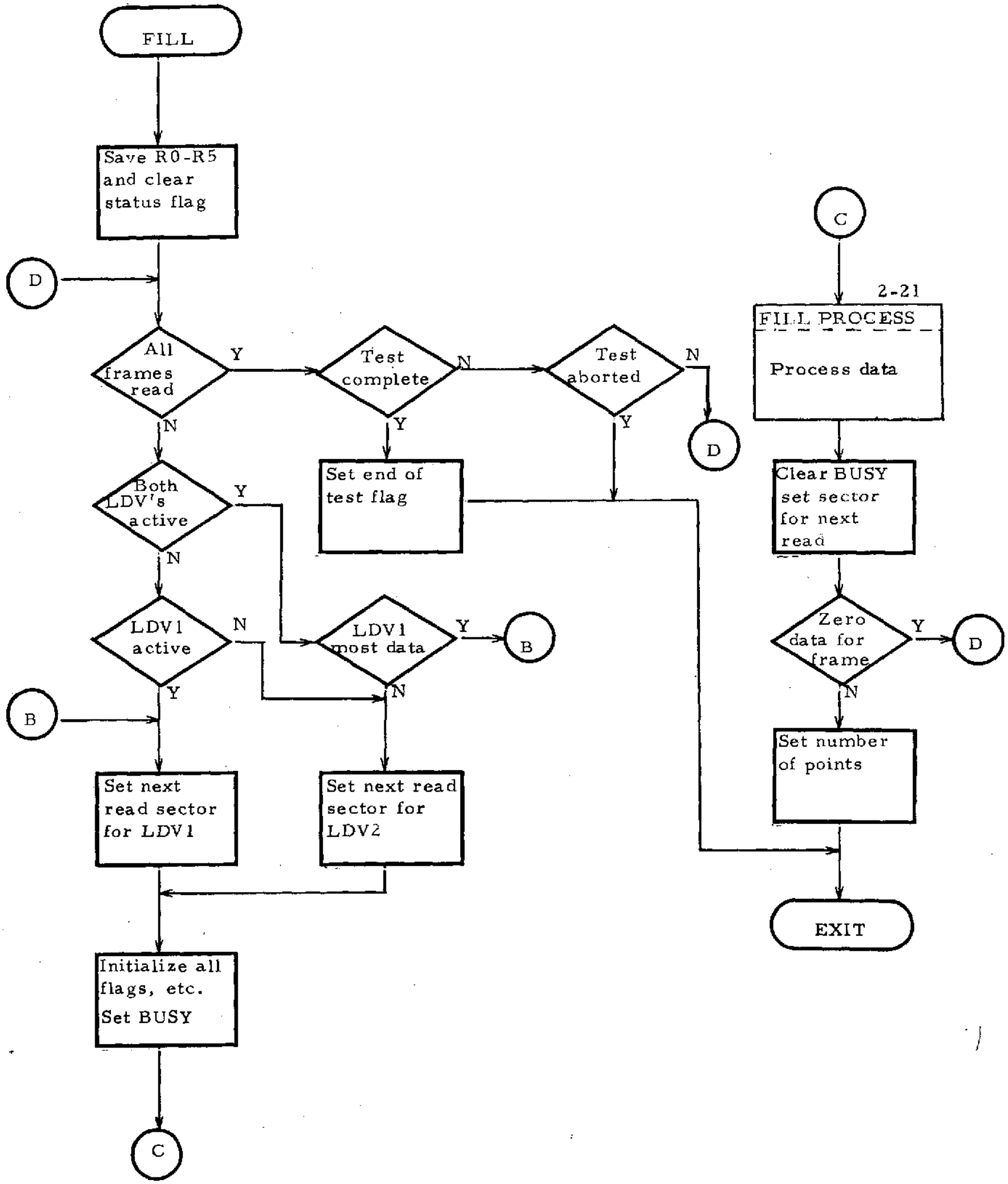

FILL
Save R0-R5 and clear status flag
D
All frames read
Y
N
Test complete
Test aborted
D
Set end of test flag
Both LDV's active
LDV1 active
LDV1 most data
B
B
Set next read sector for LDV1
Set next read sector for LDV2
Initialize all flags, etc.
Set BUSY
C
C
2-21
FILL PROCESS
Process data
Clear BUSY set sector for next read
Zero data for frame
D
Set number of points
EXIT

Figure 2-20

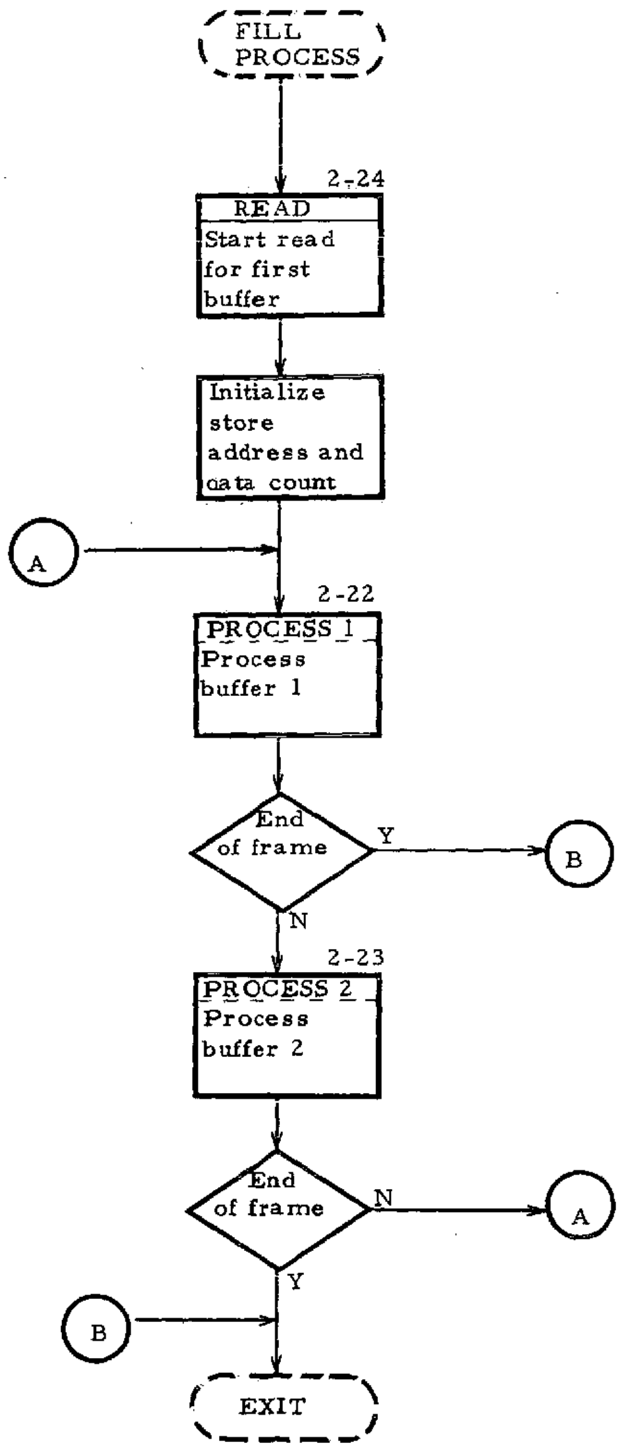

FILL PROCESS
2-24
READ
Start read for first buffer
Initialize store address and data count
A
2-22
PROCESS 1
Process buffer 1
End of frame
Y
B
N
2-23
PROCESS 2
Process buffer 2
End of frame
N
A
Y
B
EXIT

Figure 2-21

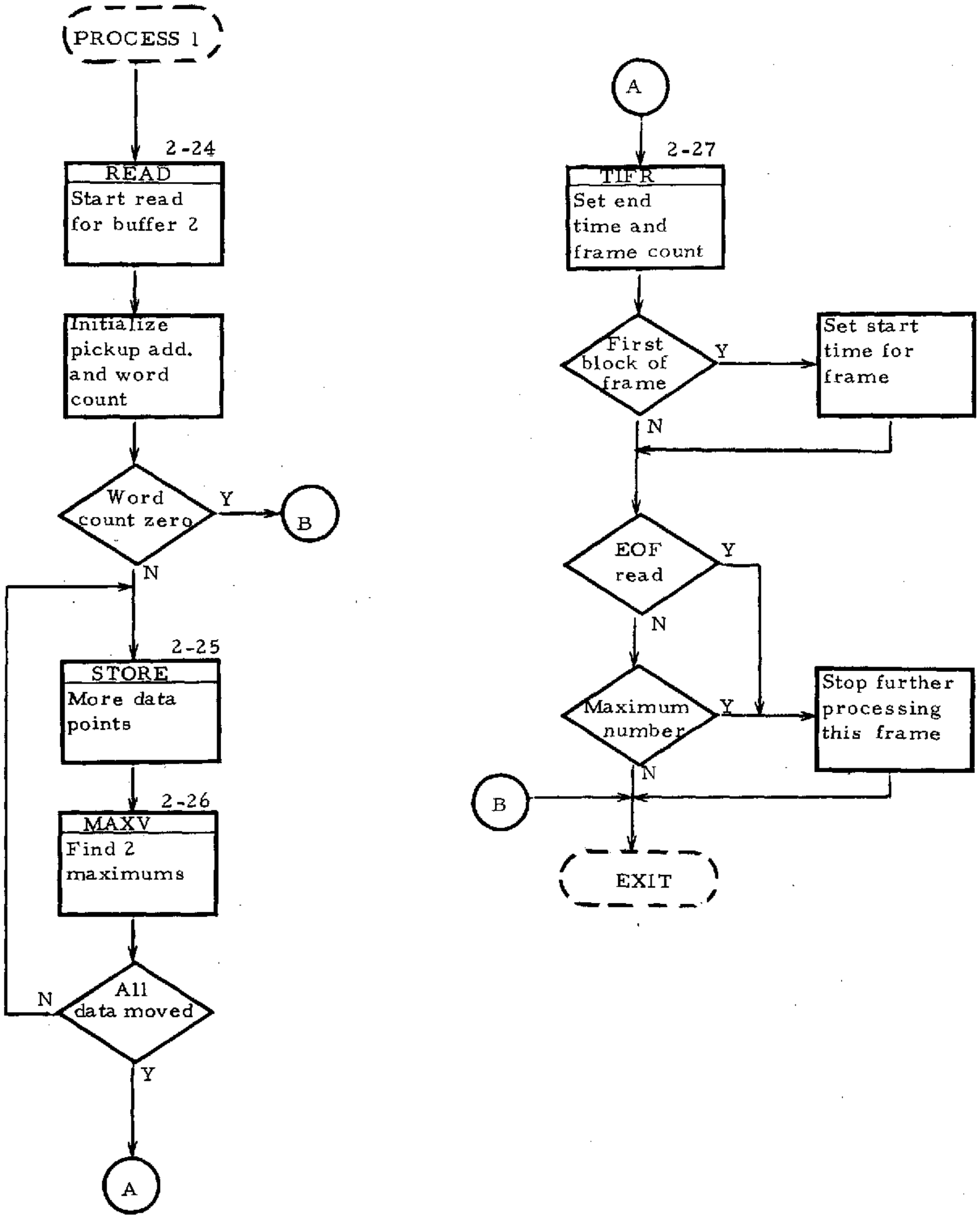

PROCESS 1
2-24
READ
Start read for buffer 2
Initialize pickup add. and word count
Word count zero
Y
B
N
2-25
STORE
More data points
2-26
MAXV
Find 2 maximums
All data moved
N
Y
A
A
2-27
TIFR
Set end time and frame count
First block of frame
Y
Set start time for frame
N
EOF read
Y
N
Maximum number
Y
Stop further processing this frame
N
B
EXIT

Figure 2-22

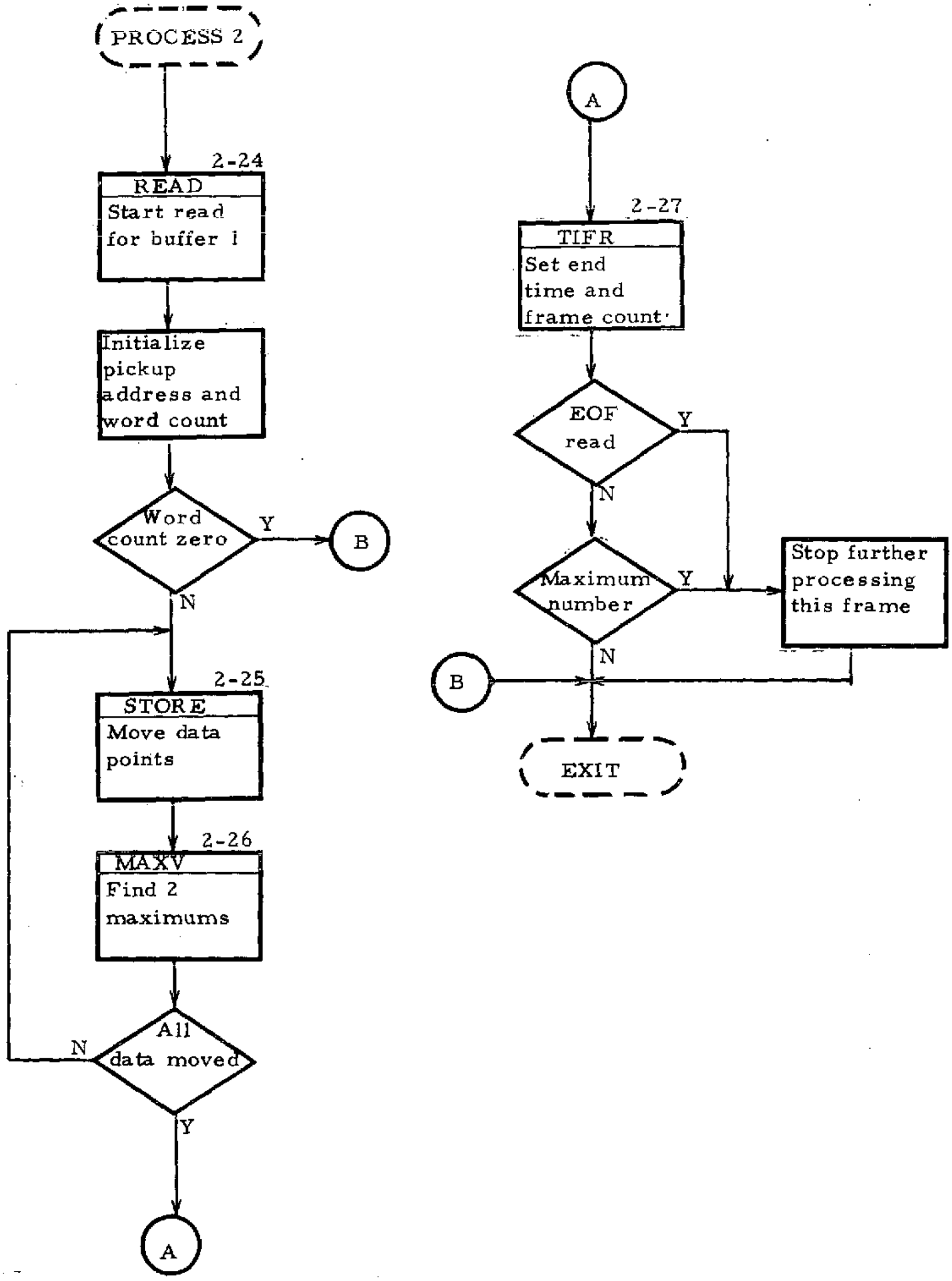

PROCESS 2
2-24
READ
Start read for buffer 1
Initialize pickup address and word count
Word count zero
Y
B
N
2-25
STORE
Move data points
2-26
MAXV
Find 2 maximums
All data moved
N
Y
A
A
2-27
TIFR
Set end time and frame count
EOF read
Y
N
Maximum number
Y
Stop further processing this frame
N
B
EXIT

Figure 2-23

Modules Required

READ

MAXV

STORE

TIFR

Critical Parameters

First word of common IFLDV denotes the test configuration as follows:

IFLD = 0, both LDV

IFLD = 1, LDV 2 only

IFLD = 2, LDV 1 only

First word in common ABTERM is used to signal a test abort:

ABTERM = 33 or 42, test is aborted

Common IHDLI is used to pass the following information:

Word 1 (MAX1) = Integer which denotes the first maximum velocity point found in data.

Word 2 (MAX2) = Integer which denotes the second maximum velocity point found in the data.

Word 3 (IEOFI) = 0, data processed normally
-1, data of frame exceeds 992 points
+1, end of flyby occurred

Common LDVDAT is used to pass the following information:

Word 1 (IFLY) = Used by FORTRAN routines.

Word 2 (IFRM) = Data frame number, where a positive frame denotes LDV1; negative, LDV 2.

Word 3 (TMINT) = Integer value of high-order time for first data word of FILL iteration. Note that each count equals 1/60 second.

Word 4 (TMINT+2) = Integer value of low-order time for first data word of FILL iteration.

Word 5 (TMEND) = Integer value of high-order time for last data word.

Word 6 (TMEND+2) = Integer value of low-order time for last data word

Word 7 (IDAY) = Used by FORTRAN routines.

Word 8 (IPLN) = Used by FORTRAN routines.

Word 9 (NUMPTS) = Integer which denotes number of data points placed in the process buffer.

Module READ

Purpose

The READ module (Figure 2-24) reads in a sector of data from the disk. For a particular FILL iteration, all reads will be performed for a particular LDV. That is, either even or odd sectors will be read.

Calling Sequence

```
JSR     PC, READ
```

(cannot be called via a FORTRAN routine).

Upon entry: R0 = 0, buffer 1 read

R0 = 2, buffer 2 read

Modules Required

LKTRAN

Module STORE

Purpose

The STORE module (Figure 2-25) extracts four data words from the input buffer and places them in the process buffer in the required format.

Calling Sequence

```
JSR     PC, STORE
```

(cannot be called via FORTRAN routine).

DISK READ

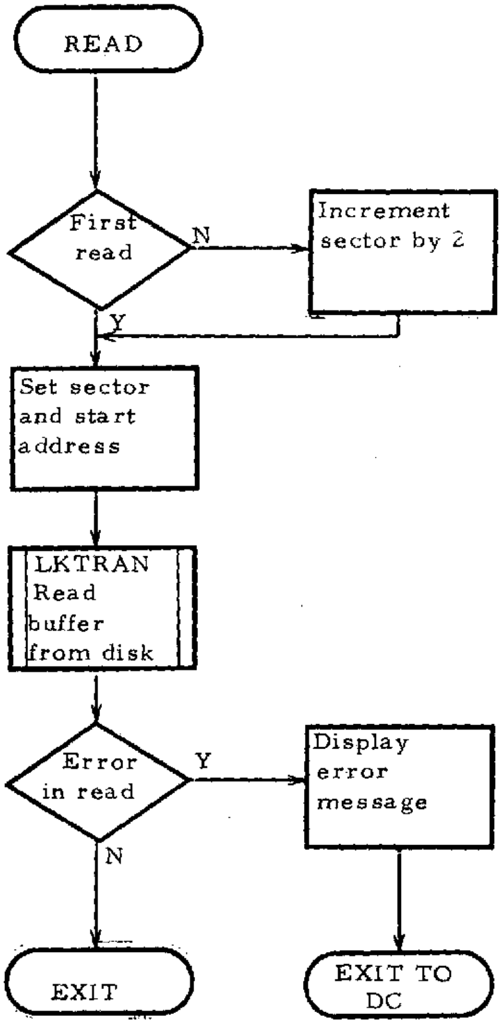


Figure 2-24

SET DATA IN PROCESS BUFFER

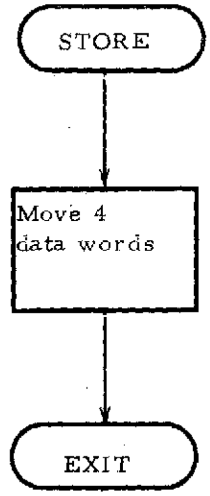


Figure 2-25

Upon entry: R2 = address of first of the four words to be moved

R4 = address of process buffer which will receive words

Upon exit: R2 = address of next four words

R4 = unchanged

Module MAXV

Purpose

The MAXV module (Figure 2-26) searches for the two maximum velocities of the FILL iteration. An integer (1-992) is set for each of the maximums to denote the data point numbers for the process buffer of the current FILL iteration.

Calling Sequence

```
JSR     PC, MAXV
```

(cannot be called via FORTRAN routine).

Upon entry: R4 = address of current velocity in process buffer

R1 = current data point number

Upon exit: R4 = unchanged

R1 = unchanged

Module TIFR

Purpose

The TIFR module (Figure 2-27) sets the last time and the frame number in the designated COMMON (see module FILL).

Calling Sequence

```
JSR     PC, TIFR
```

(cannot be called via FORTRAN routine).

Upon entry: R2 = current buffer address

Upon exit: R2 = unchanged

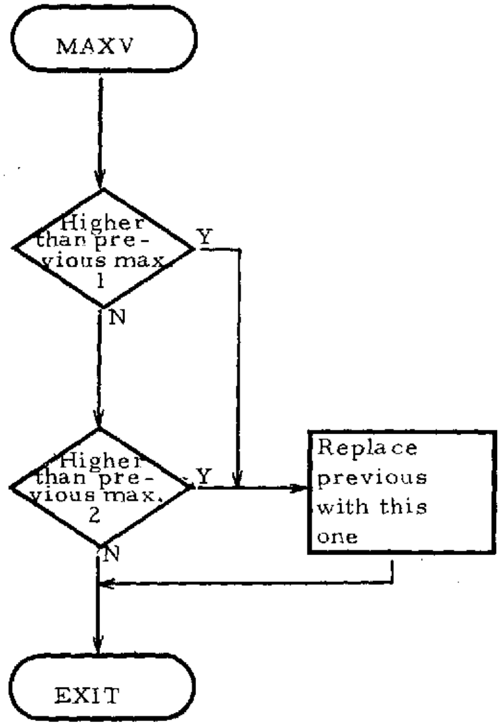


Figure 2-26

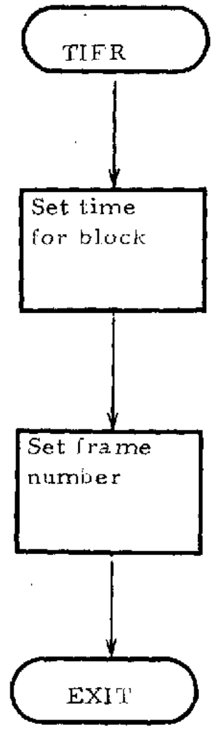


Figure 2-27

3. FORTRAN ROUTINES

This section describes the routines which initialize variables, locate vortex centers, and provide interfaces between the data acquisition routines, the vortex location routines, and the display controller. These routines are written in FORTRAN to facilitate program modification as more knowledge about vortex behavior and detection is gained.

3.1 Start Flyby

Name

STRT

Calling Sequence

CALL STRT (N)

Where N is a six element array with the elements containing the following:

- N(1) contains the number of the display that was on the screen when STRT was called.
- N(2) contains the option number which was selected from the display.
- N(3) contains the number of bytes in a compose field.
- N(4) contains a flag that indicates type of input field.
- N(5) and N(6) contain data from the input field.

Description of Function

Subroutine STRT (Figures 3-1 through 3-12) provides the interface between the assembly language I/O routines, the vortex location algorithm, and the display controller. The routine initializes the output data tape, directs the placement of the chosen ouput display backgrounds on the terminal screens, directs the filling of data buffers from either real time LDV input or from tape, directs output of real time data to tape, and calls the vortex location algorithm. STRT also handles flyby termination as directed by LDV input signals, end of files on input data tapes, or operator keyboard input signals and cycles the program to the appropriate starting point for the next data input.

External References

DISPIO, RTV, OPEN, SFUN, INIT1, SETAD, DFLT, SCAT, PTYP, DISAB1, DISABL, FILL, PUT, VREAD, CENTRD, and WAIT.

START FLYBY

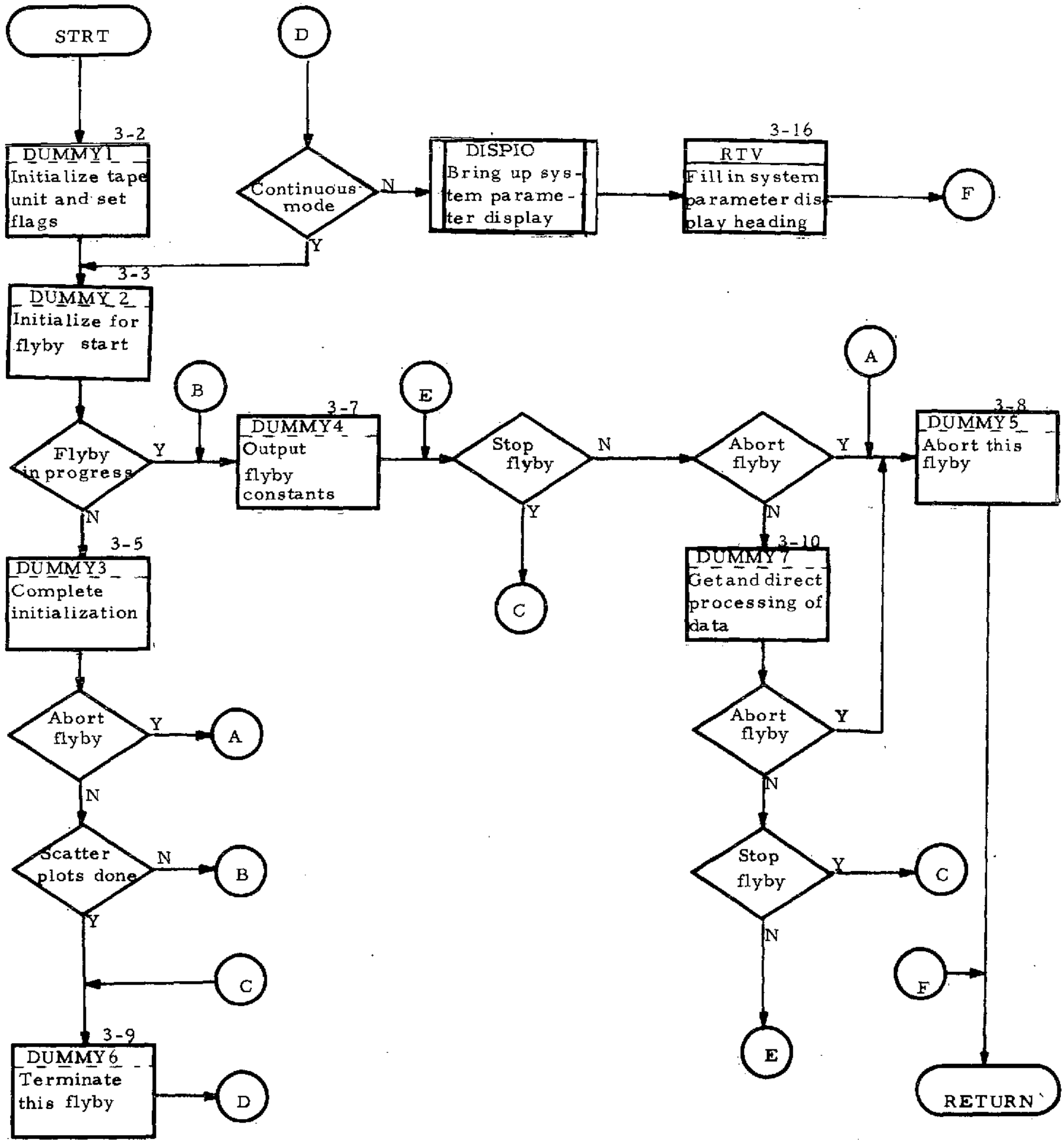


Figure 3-1

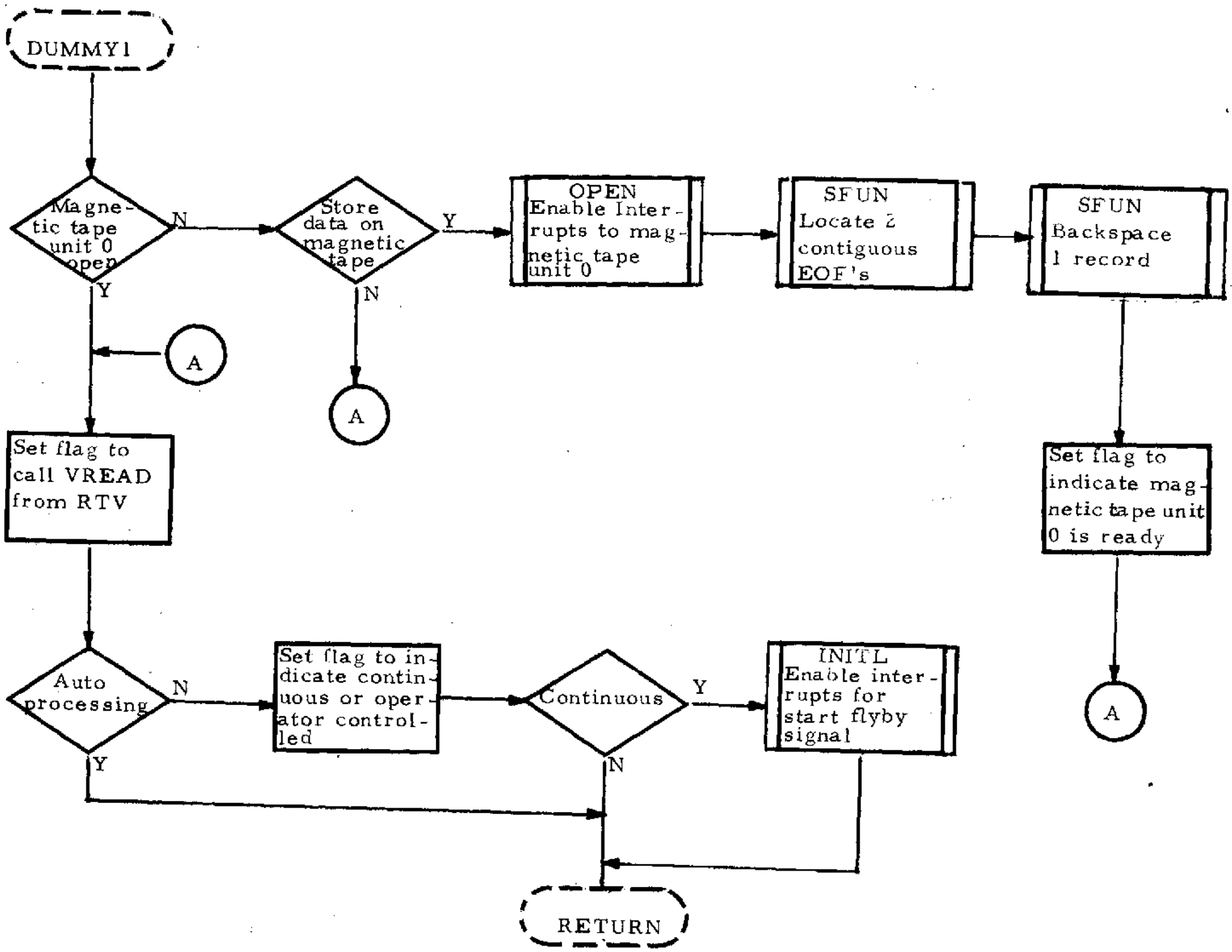

DUMMY1
Magnetic tape unit 0 open
N
Y
Store data on magnetic tape
Y
N
A
OPEN
Enable Interrupts to magnetic tape unit 0
SFUN
Locate 2 contiguous EOF's
SFUN
Backspace 1 record
Set flag to call VREAD from RTV
Set flag to indicate magnetic tape unit 0 is ready
Auto processing
N
Y
Set flag to indicate continuous or operator controlled
Continuous
Y
N
INITL
Enable interrupts for start flyby signal
RETURN

Figure 3-2

FLYBY INITIALIZATION

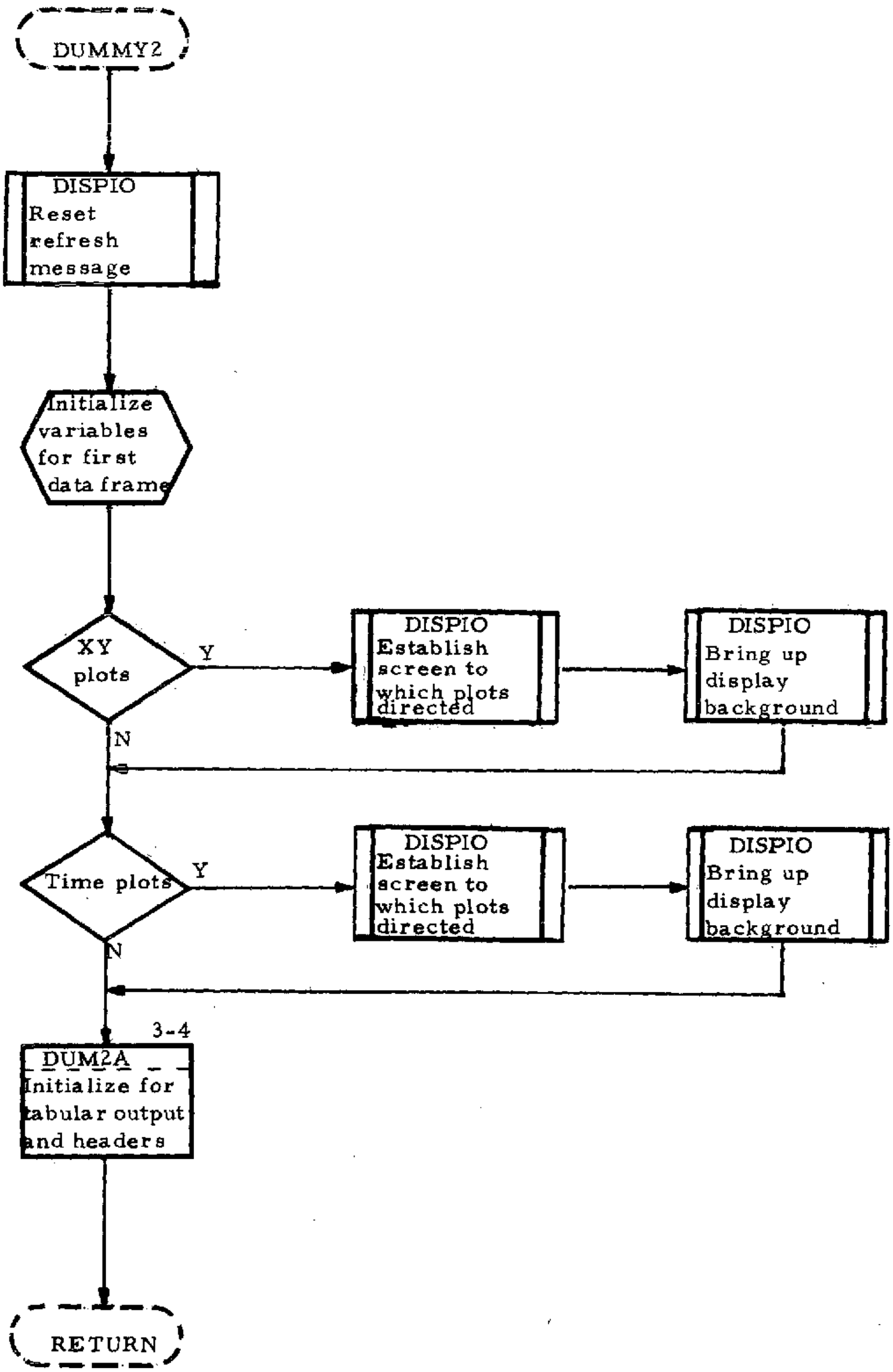


Figure 3-3

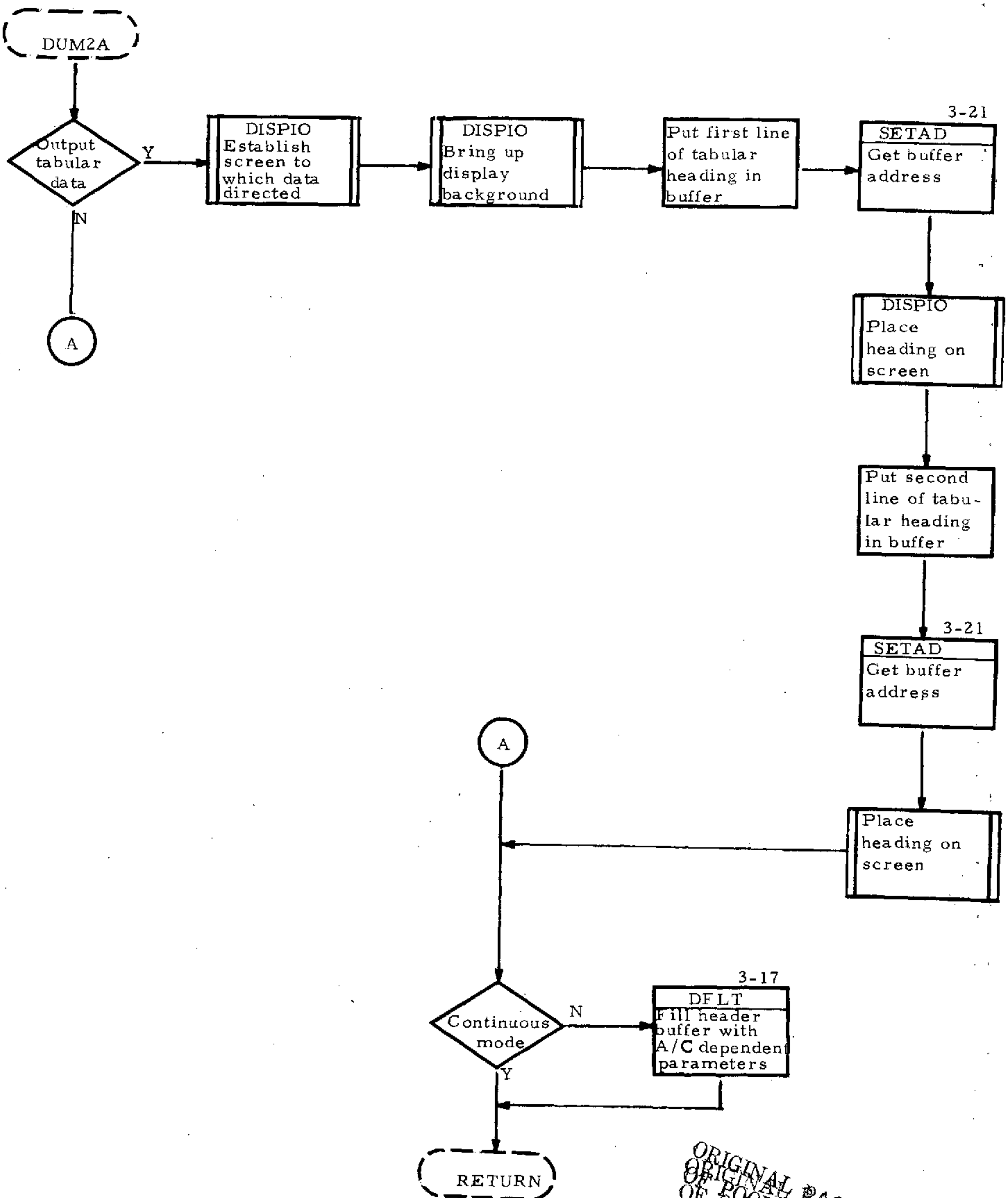

DUM2A
Output tabular data
Y
N
A
DISPIO
Establish screen to which data directed
DISPIO
Bring up display background
Put first line of tabular heading in buffer
3-21
SETAD
Get buffer address
DISPIO
Place heading on screen
Put second line of tabular heading in buffer
3-21
SETAD
Get buffer address
A
Place heading on screen
Continuous mode
N
Y
3-17
DFLT
Fill header buffer with A/C dependent parameters
RETURN

Figure 3-4

COMPLETE INITIALIZATION FOR FLYBY

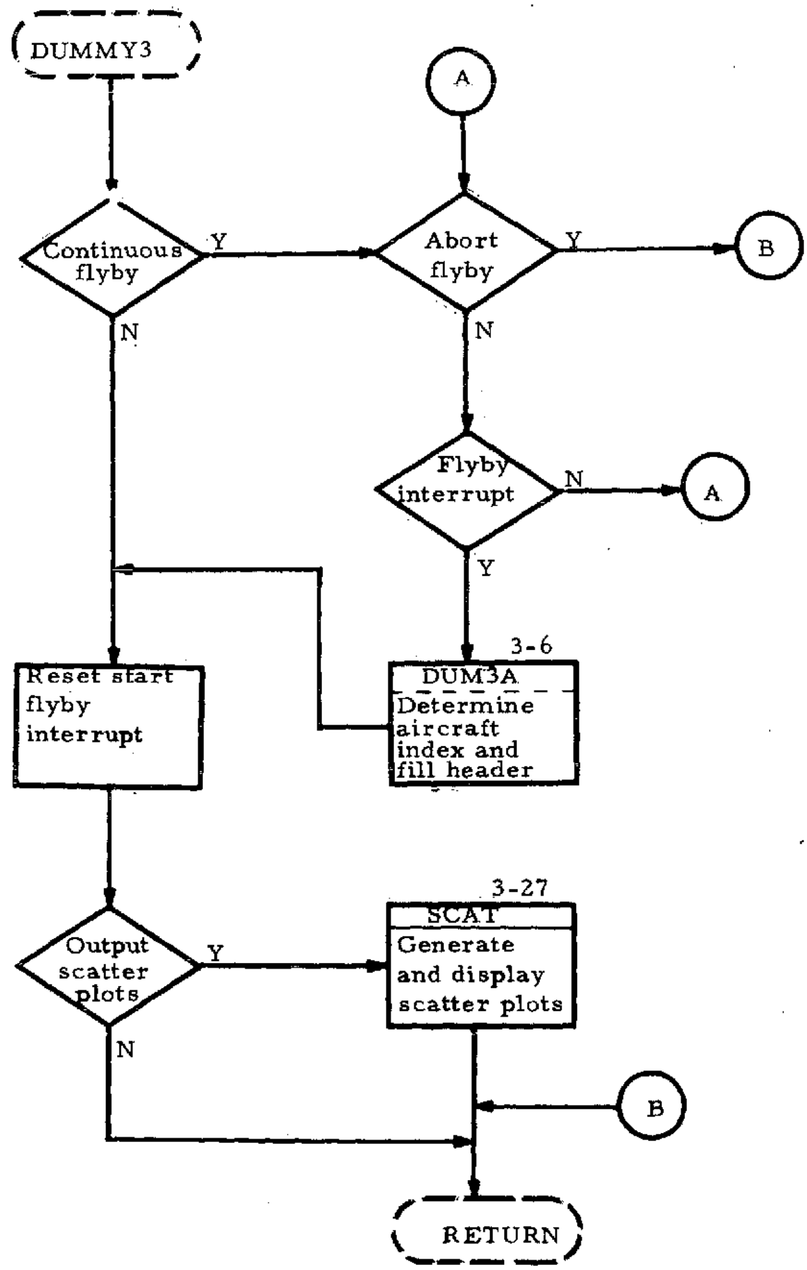


Figure 3-5

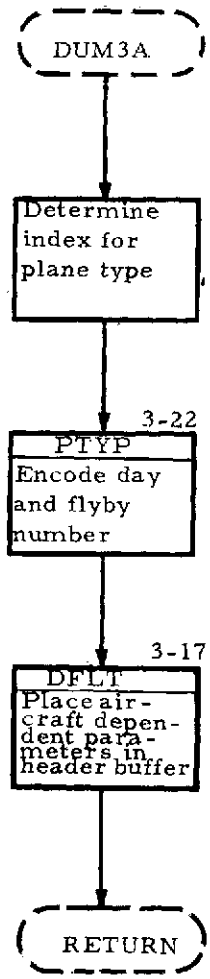

DUM3A
Determine index for plane type
3-22
PTYP
Encode day and flyby number
3-17
DFLT
Place aircraft dependent parameters in header buffer
RETURN

Figure 3-6

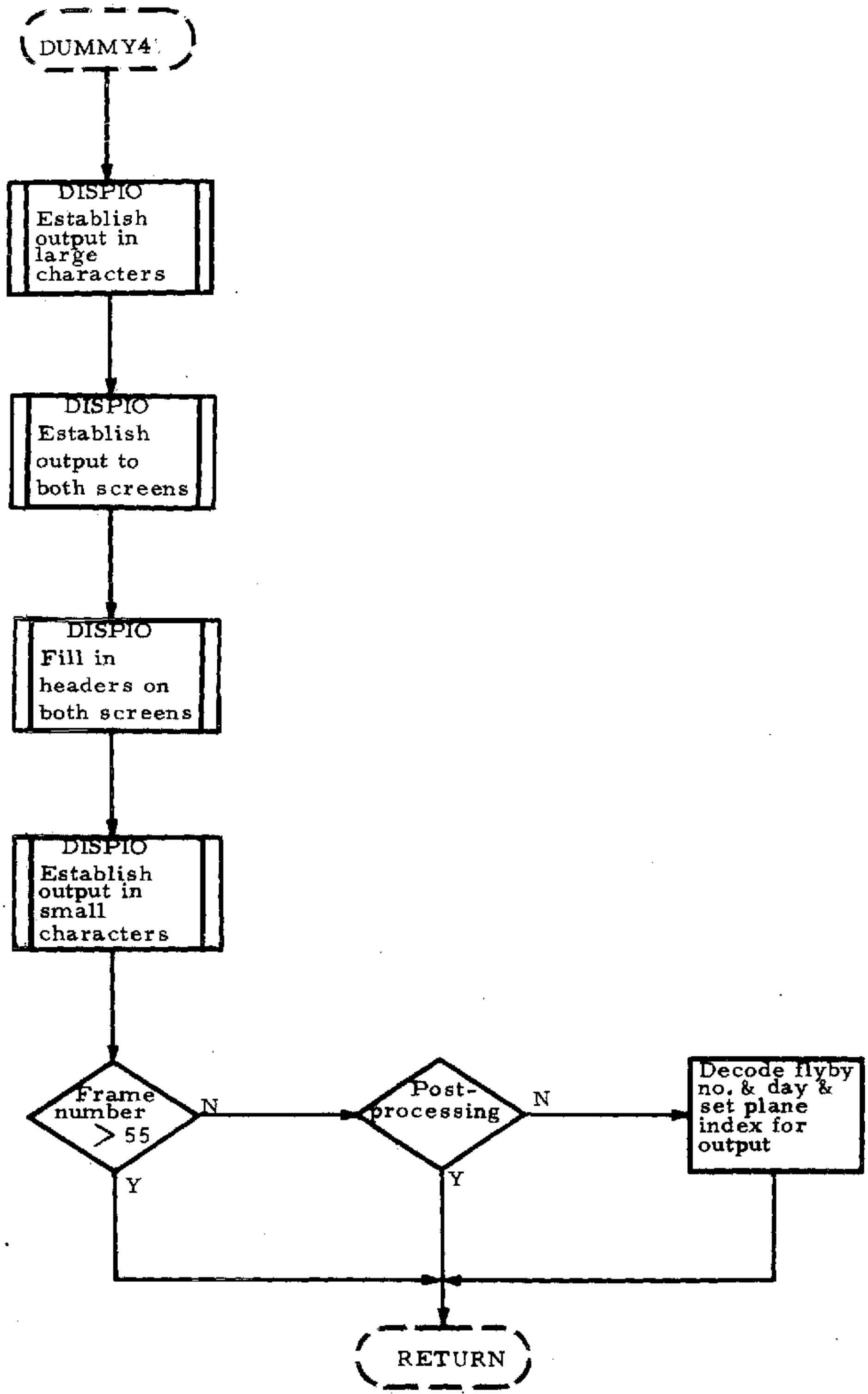

DUMMY4
DISPIO
Establish output in large characters
DISPIO
Establish output to both screens
DISPIO
Fill in headers on both screens
DISPIO
Establish output in small characters
Frame number > 55
N
Y
Post-processing
N
Y
Decode flyby no. & day & set plane index for output
RETURN

Figure 3-7

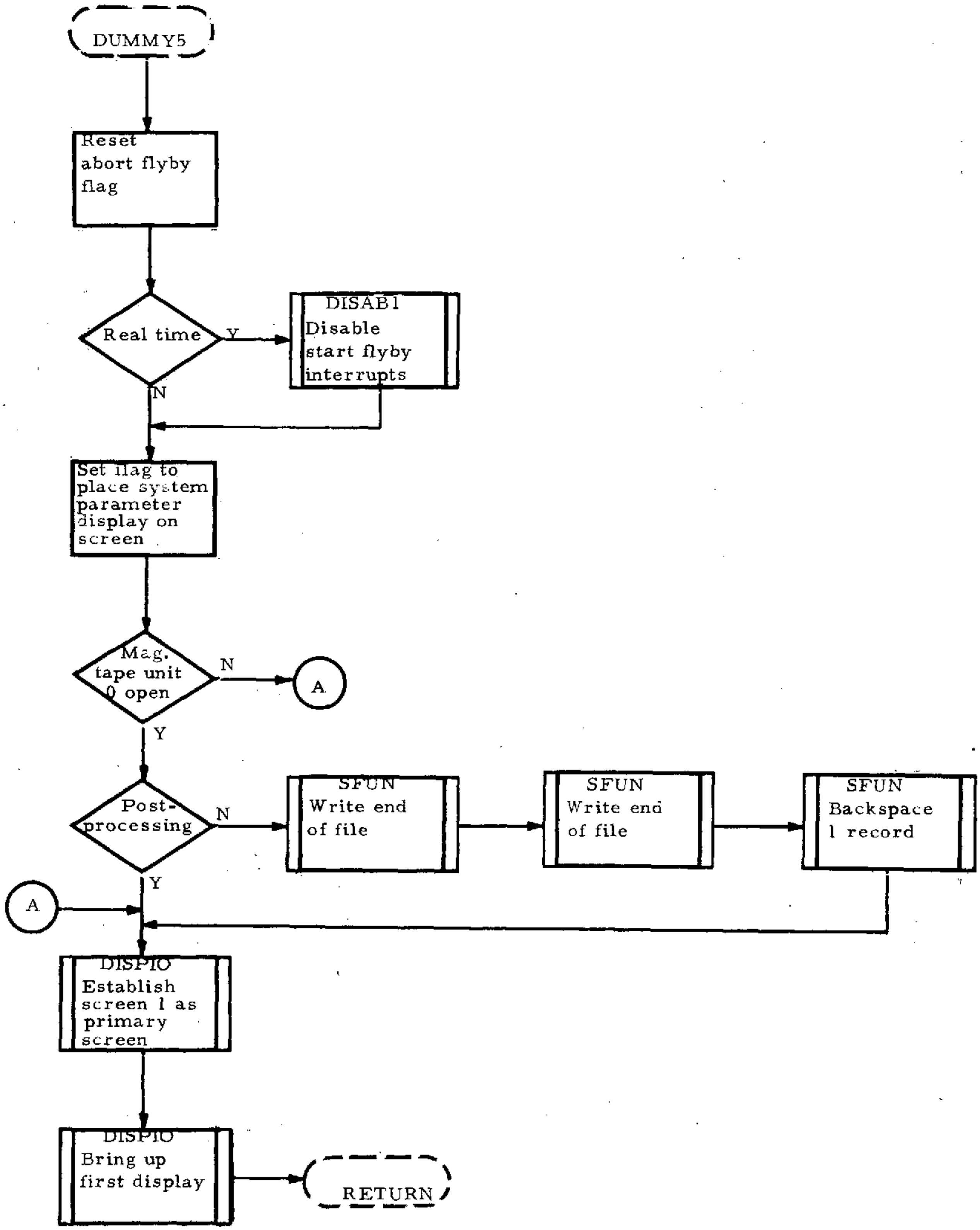

DUMMY5
Reset abort flyby flag
Real time
Y
N
DISAB1
Disable start flyby interrupts
Set flag to place system parameter display on screen
Mag. tape unit 0 open
N
A
Y
Post-processing
N
SFUN
Write end of file
SFUN
Write end of file
SFUN
Backspace 1 record
Y
A
DISPIO
Establish screen 1 as primary screen
DISPIO
Bring up first display
RETURN

Figure 3-8

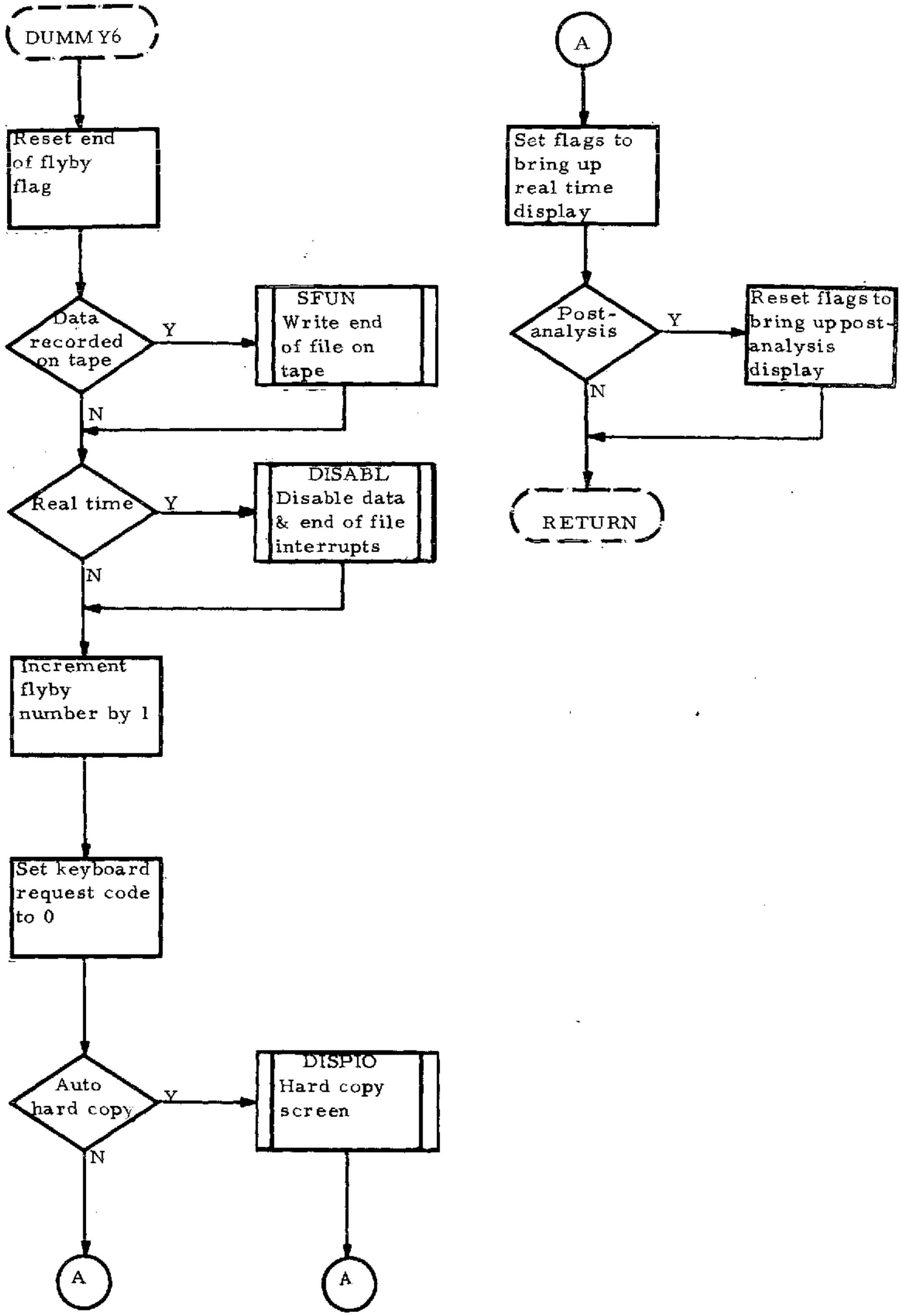
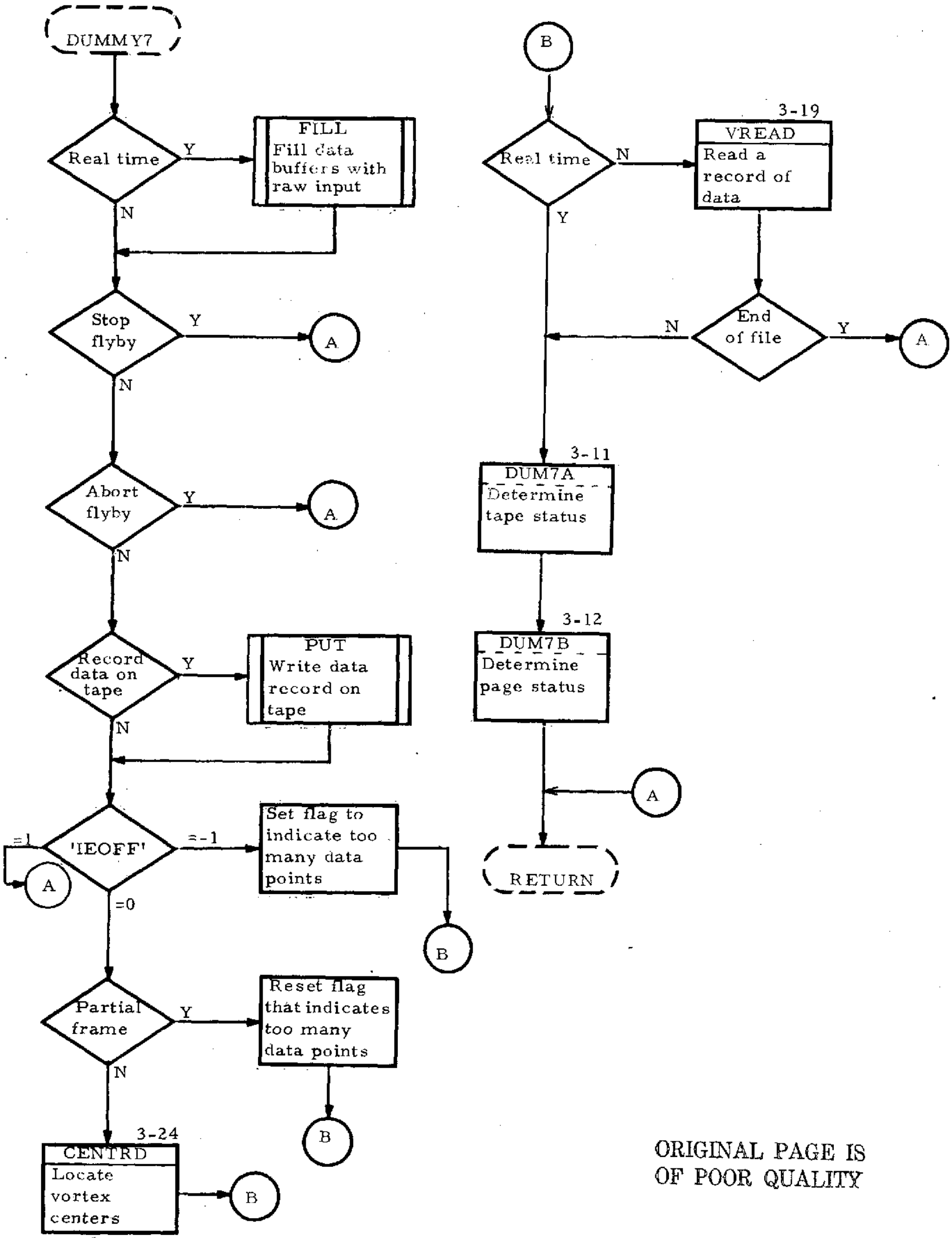

DUMMY6
Reset end of flyby flag
Data recorded on tape
Y
SFUN
Write end of file on tape
N
Real time
Y
DISABL
Disable data & end of file interrupts
N
Increment flyby number by 1
Set keyboard request code to 0
Auto hard copy
Y
DISPIO
Hard copy screen
N
A
A
A
Set flags to bring up real time display
Post-analysis
Y
Reset flags to bring up post-analysis display
N
RETURN

Figure 3-9



ORIGINAL PAGE IS OF POOR QUALITY

Figure 3-10

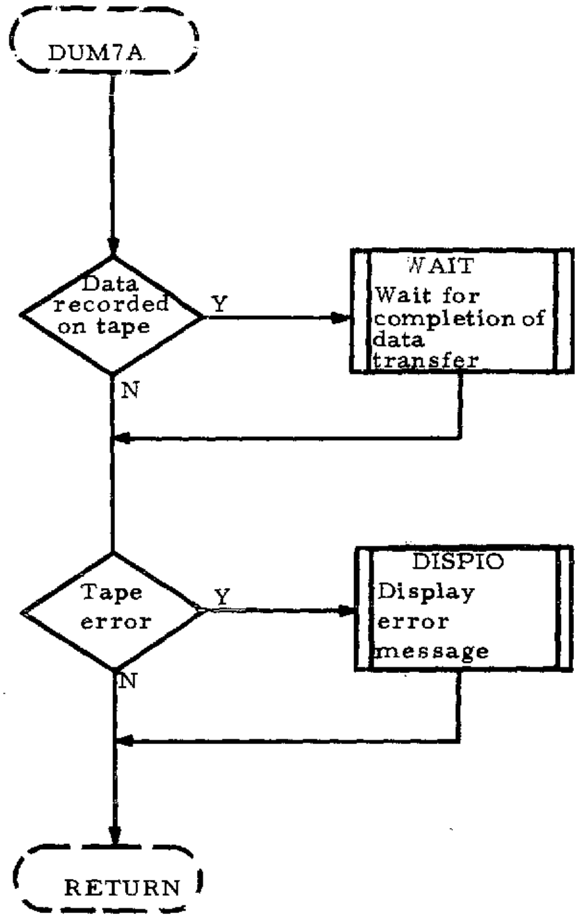

DUM7A
Data recorded on tape
Y
WAIT
Wait for completion of data transfer
N
Tape error
Y
DISPIO
Display error message
N
RETURN

Figure 3-11

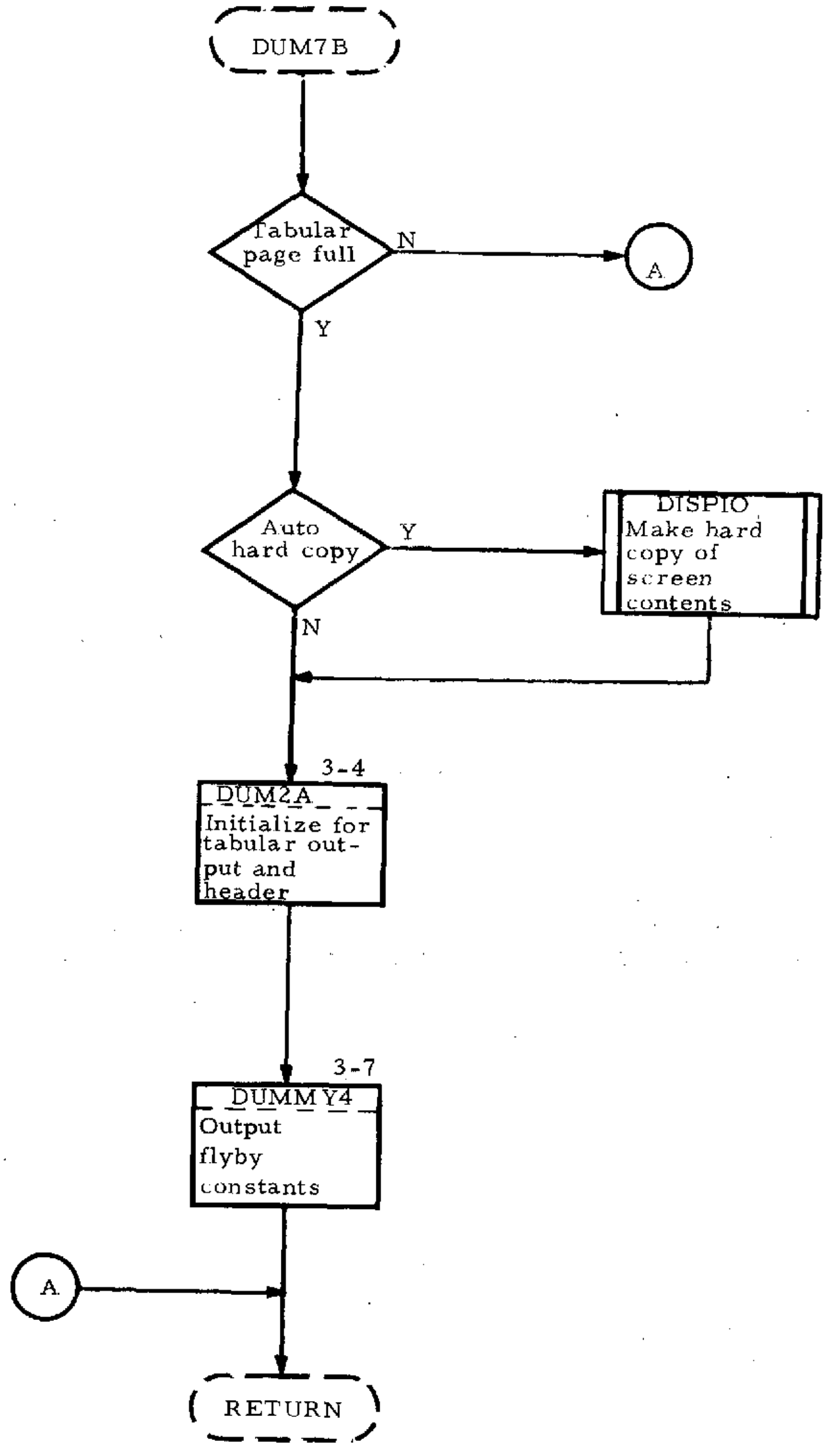

DUM7B
Tabular page full
N
A
Y
Auto hard copy
Y
DISPIO
Make hard copy of screen contents
N
3-4
DUM2A
Initialize for tabular output and header
3-7
DUMMY4
Output flyby constants
A
RETURN

Figure 3-12

3.2 Device Selection

Name

DSEL

Calling Sequence

CALL DSEL (N)

Where N is an eight element array with the elements containing the following:

- N(1) contains the number of the display that was on the screen when DSEL was called.
- N(2) contains the option number which was selected from the display.
- N(3) contains the number of bytes in a compose field.
- N(4) contains a flag indicating the type of input field.
- N(5) through N(8) contain input field values.

Description of Function

Subroutine DSEL (Figures 3-13 and 3-14) determines which input and output devices were selected by the operator and sets flags to indicate the chosen devices. The routine also is used to place compose field input for flyby number and day in header buffers for displays. In post-analysis, the routine searches the data tape for flyby numbers or day numbers that match the input number.

External References

SFUN, DISPIO, PTYP, and VREAD.

3.3 Get Addresses for Output

Name

ONCE

Calling Sequence

CALL ONCE

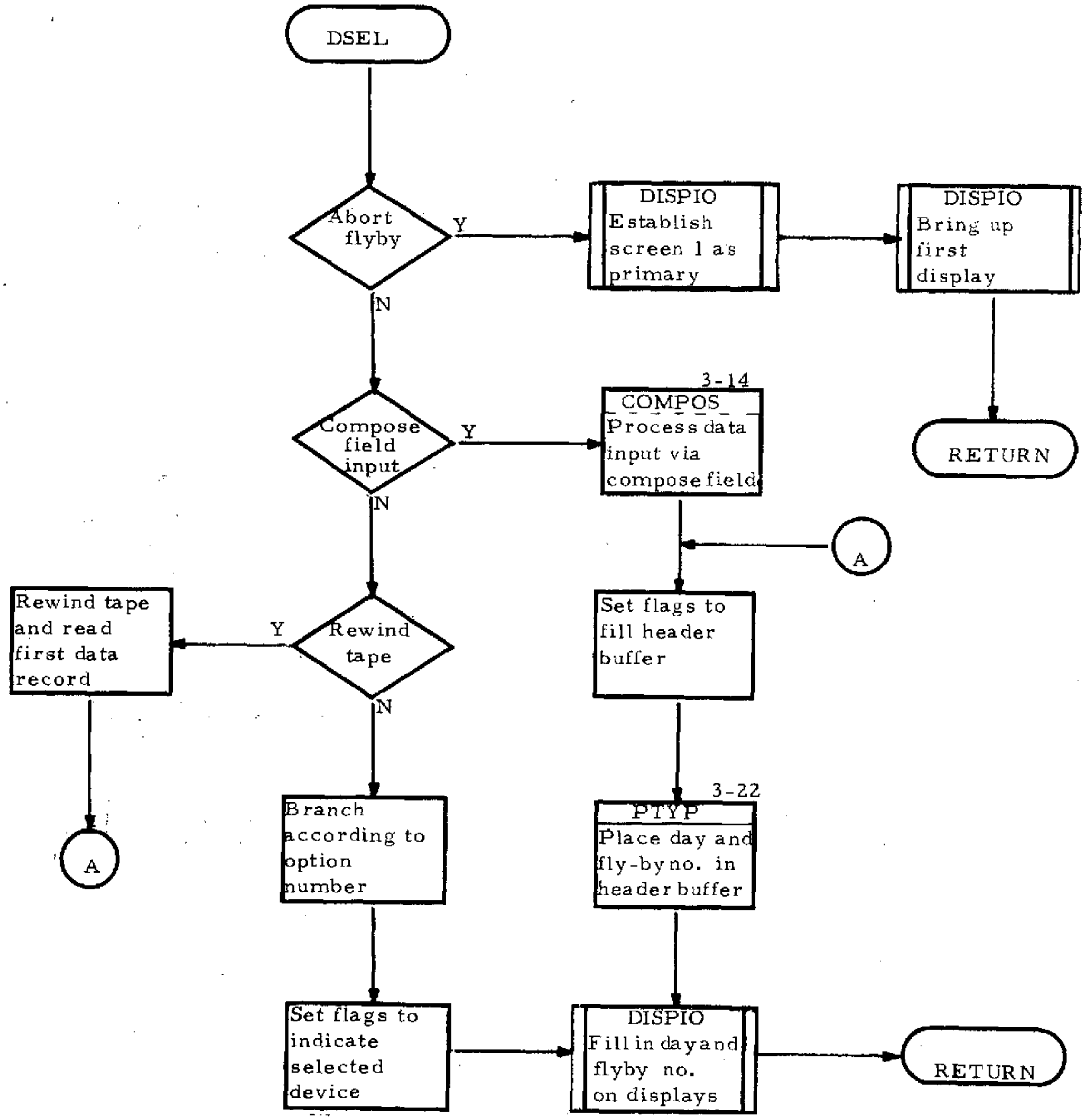

DSEL
Abort flyby
Y
N
DISPIO
Establish screen 1 as primary
DISPIO
Bring up first display
RETURN
Compose field input
Y
N
3-14
COMPOS
Process data input via compose field
A
Set flags to fill header buffer
Rewind tape
Y
N
Rewind tape and read first data record
A
Branch according to option number
3-22
PTYP
Place day and fly-by no. in header buffer
Set flags to indicate selected device
DISPIO
Fill in day and flyby no. on displays
RETURN

Figure 3-13

COMPOSE FIELD INPUTS FOR DAY AND FLYBY NUMBER

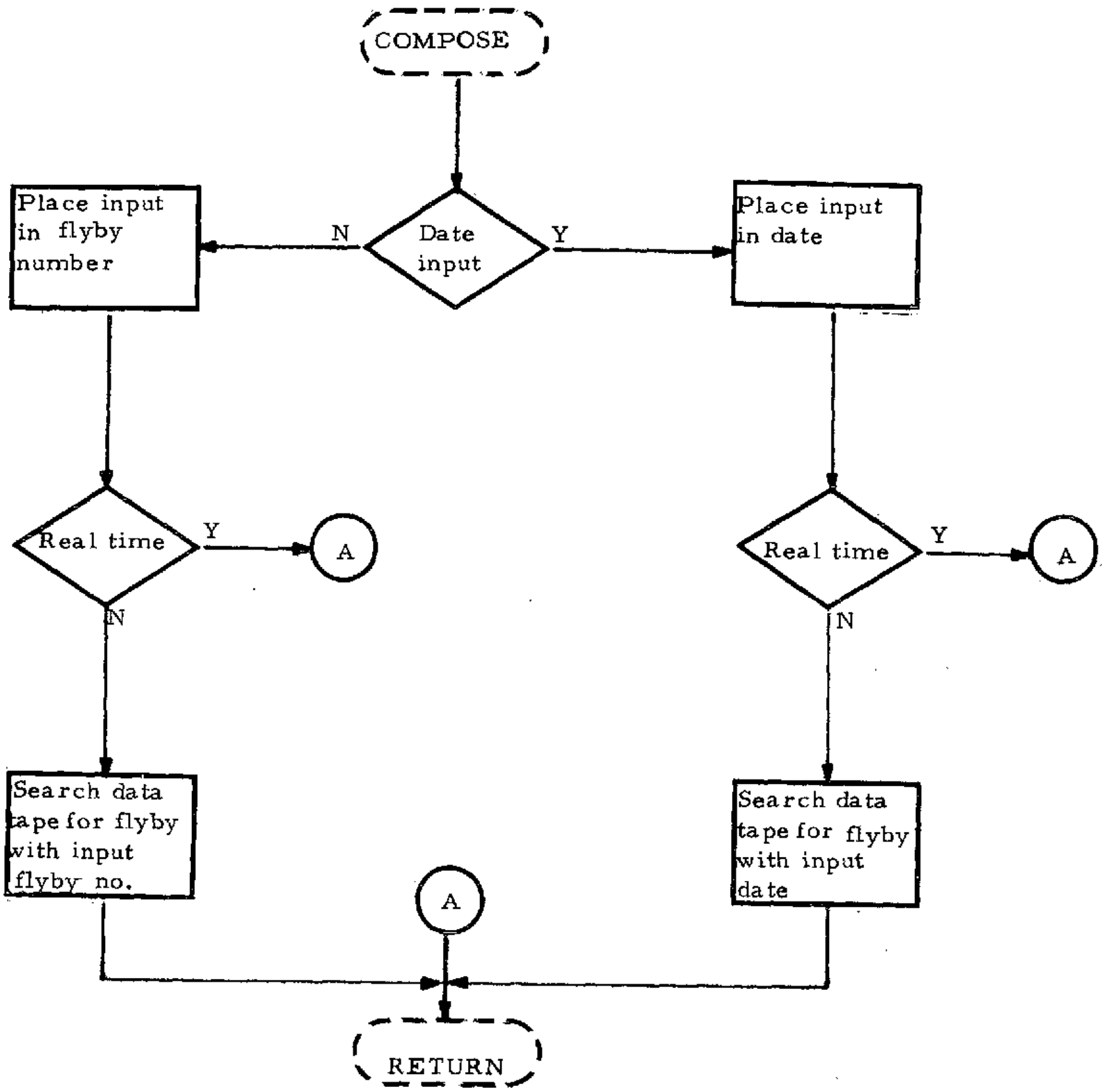


Figure 3-14

Description of Function

Subroutine ONCE (Figure 3-15) is entered only when the program is loaded. It calls the display controller to establish high-speed output in large characters and calls SETAD to get the addresses of the variables which follow:

- o Blank.
- o XYSYM odd elements which contain the letters of the alphabet.
- o PLTSI which contains an *.
- o PLTS3 which contains an o.
- o IDATA which is an output buffer array.
- o IL1 which is the buffer that contains dit positions for write-through lines showing default values for system parameters.
- o IL2 which is the buffer that contains dit positions for write-through lines showing default values for aircraft dependent parameters.
- o DATE which contains flyby date.
- o IFLB which contains flyby number.

It also converts the default correlation radius from feet to counts.

External References

DISPIO and SETAD.

3.4 System Initialization

Name

RTV

Calling Sequence

CALL RTV (N)

Where N is a six element array with the elements containing the following:

- o N(1) contains the number of the display that was on the screen when RTV was called.

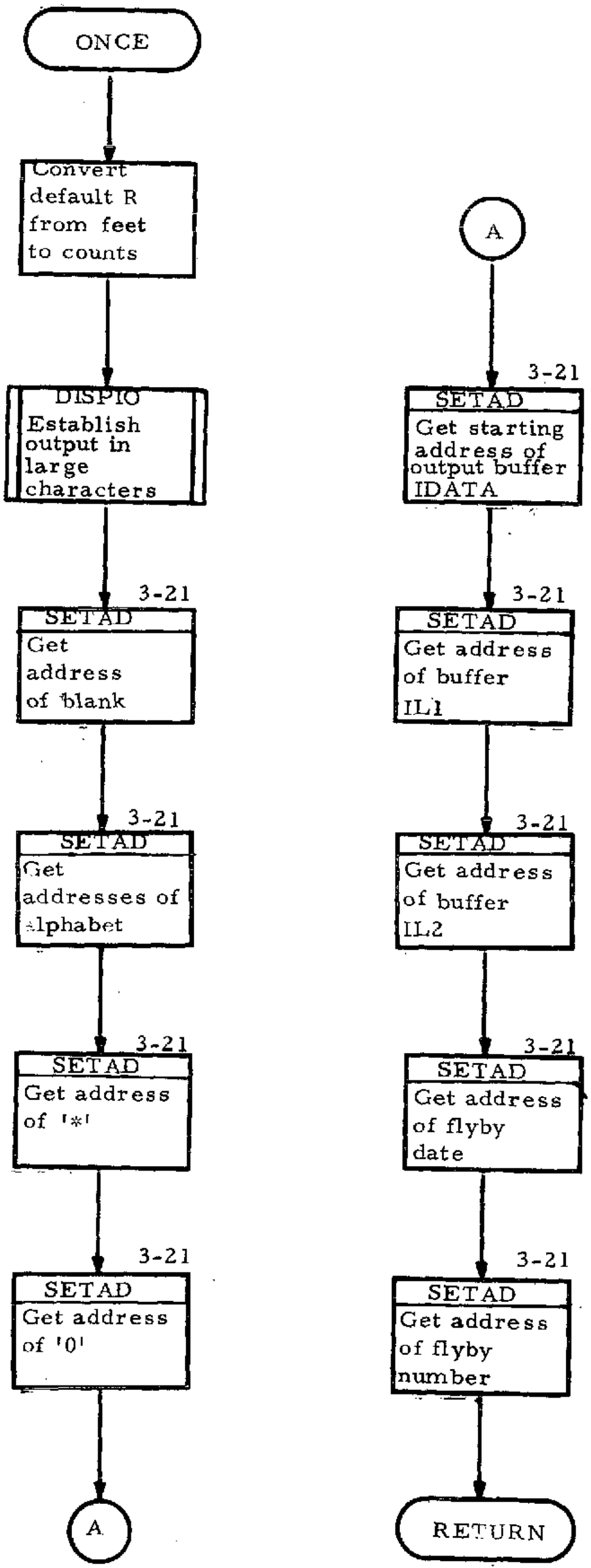

ONCE
Convert default R from feet to counts
DISPIO
Establish output in large characters
3-21
SETAD
Get address of blank
3-21
SETAD
Get addresses of alphabet
3-21
SETAD
Get address of '*'
3-21
SETAD
Get address of '0'
A
A
3-21
SETAD
Get starting address of output buffer IDATA
3-21
SETAD
Get address of buffer IL1
3-21
SETAD
Get address of buffer IL2
3-21
SETAD
Get address of flyby date
3-21
SETAD
Get address of flyby number
RETURN

Figure 3-15

- o N(2) contains the number of the option which was selected from the display.
- o N(3) contains the number of bytes in a compose field.
- o N(4) contains a flag which indicates the type of input field.
- o N(5) and N(6) contain data from the input field.

On any given call, all of the elements of N may not be defined.

Description of Function

During Post-Analysis, RTV (Figure 3-16) enables the interface to magnetic tape unit 1 and reads the first data record for each flyby. During both post-analysis and real-time operation, the routine sets the coordinates and calls the display controller to draw write-through lines indicating chosen system parameters. It also places the day and flyby number on the system parameters displays.

External References

DISPIO, OPEN, SFUN, VREAD, and PTYP.

3.5 Update System Parameters

Name

DFLT

Calling Sequence

CALL DFLT (N)

Where N is an eight element array containing the following:

- o N(1) contains the number of the display that was on the screen when DFLT was called.
- o N(2) contains the option number that was selected from the display.
- o N(3) contains the number of bytes in a compose field.
- o N(4) contains a flag indicating the type of input field.
- o N(5) through N(8) contain input field values.

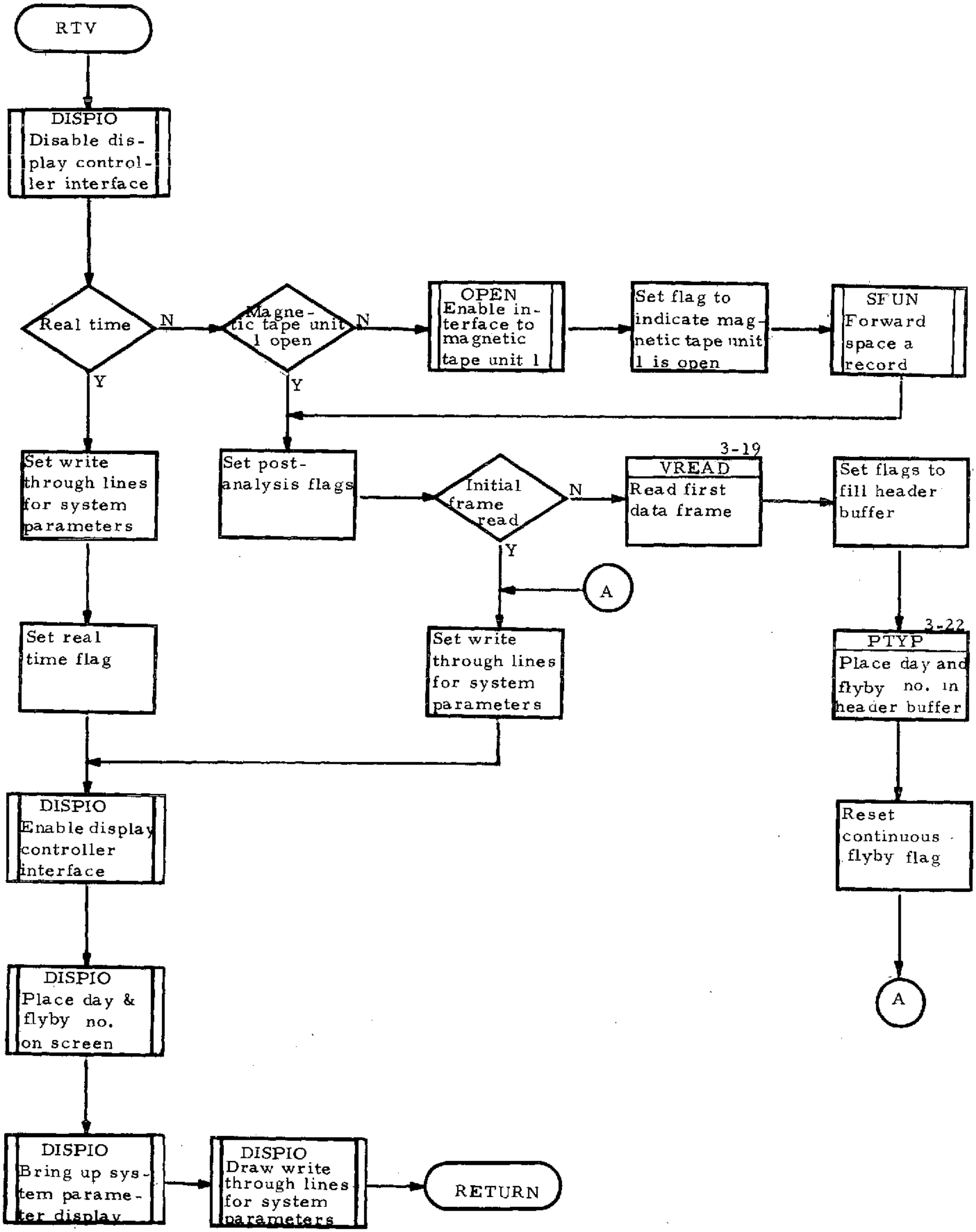

RTV
DISPIO
Disable display controller interface
Real time
N
Y
Magnetic tape unit 1 open
N
Y
OPEN
Enable interface to magnetic tape unit 1
Set flag to indicate magnetic tape unit 1 is open
SFUN
Forward space a record
Set write through lines for system parameters
Set post-analysis flags
Initial frame read
N
Y
3-19
VREAD
Read first data frame
Set flags to fill header buffer
A
Set real time flag
Set write through lines for system parameters
3-22
PTYP
Place day and flyby no. in header buffer
Reset continuous flyby flag
A
DISPIO
Enable display controller interface
DISPIO
Place day & flyby no. on screen
DISPIO
Bring up system parameter display
DISPIO
Draw write through lines for system parameters
RETURN

Figure 3-16

Description of Function

Subroutine DFLT (Figure 3-17) changes the aircraft dependent parameters from a default value to a value input through the keyboard and if the update default option was selected, it updates the default value for the particular aircraft to the input value. The routine also sets the buffer containing the heading that displays aircraft dependent parameters.

External References

None.

3.6 Raw Data Dump

Name

DBUG

Calling Sequence

CALL DBUG

Description of Function

Subroutine DBUG (Figure 3-18) gives a dump of the raw data in counts that is received from the LDV's. The data includes the horizontal and vertical coordinates of a point, the number of filters, the intensity, the peak velocity, and the velocity at maximum intensity.

External References

SSWTCH, DISPIO, and SUBBIT.

3.7 Read Data From Tape

Name

VREAD

Calling Sequence

CALL VREAD

Description of Function

Subroutine VREAD (Figure 3-19) obtains raw position, velocity, and intensity data from tape for post processing and prepares the data for processing.

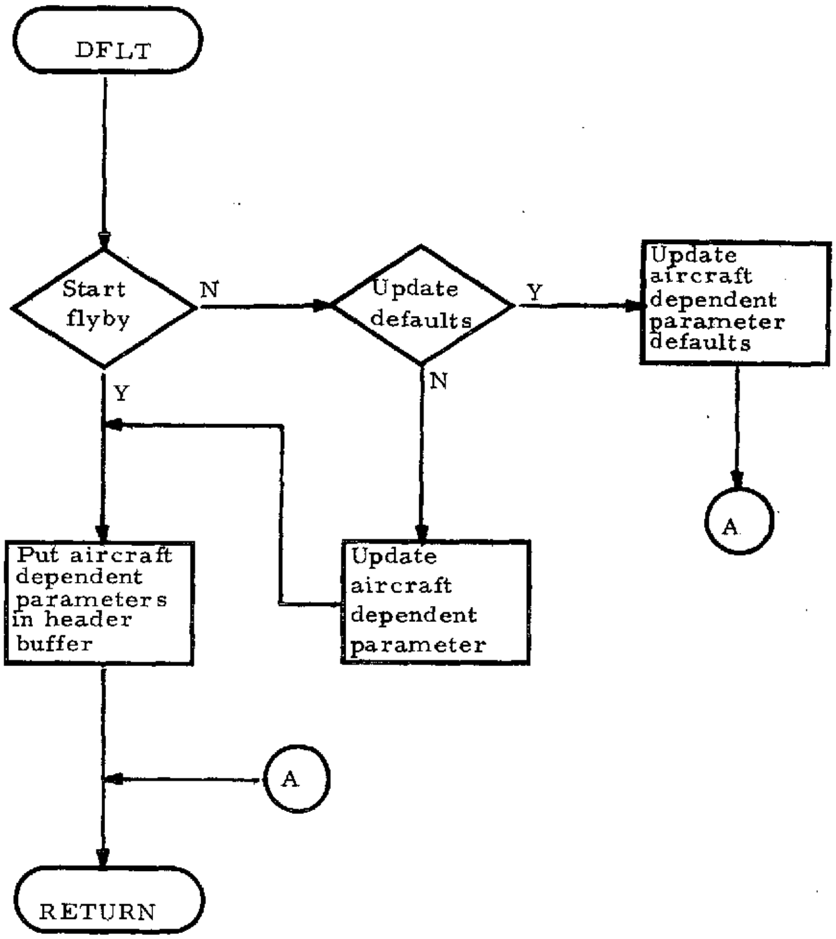

DFLT
Start flyby
N
Update defaults
Y
Update aircraft dependent parameter defaults
N
Y
A
Put aircraft dependent parameters in header buffer
Update aircraft dependent parameter
A
RETURN

Figure 3-17

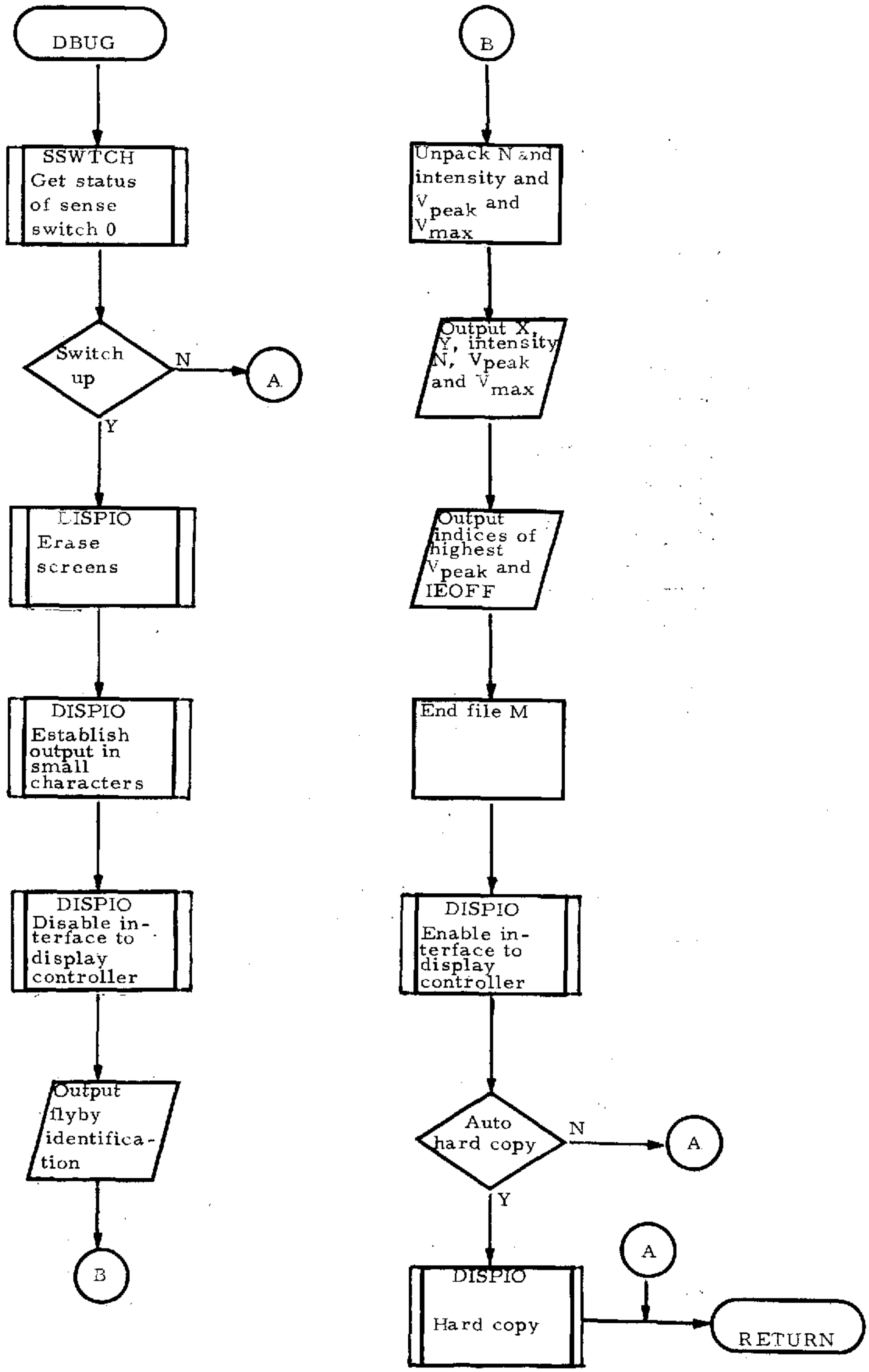

DBUG
SSWTCH Get status of sense switch 0
Switch up
N
A
Y
DISPIO Erase screens
DISPIO Establish output in small characters
DISPIO Disable interface to display controller
Output flyby identification
B
B
Unpack N and intensity and V_{peak} and V_{max}
Output X, Y, intensity N, V_{peak} and V_{max}
Output indices of highest V_{peak} and IEOFF
End file M
DISPIO Enable interface to display controller
Auto hard copy
N
A
Y
A
DISPIO Hard copy
RETURN

Figure 3-18

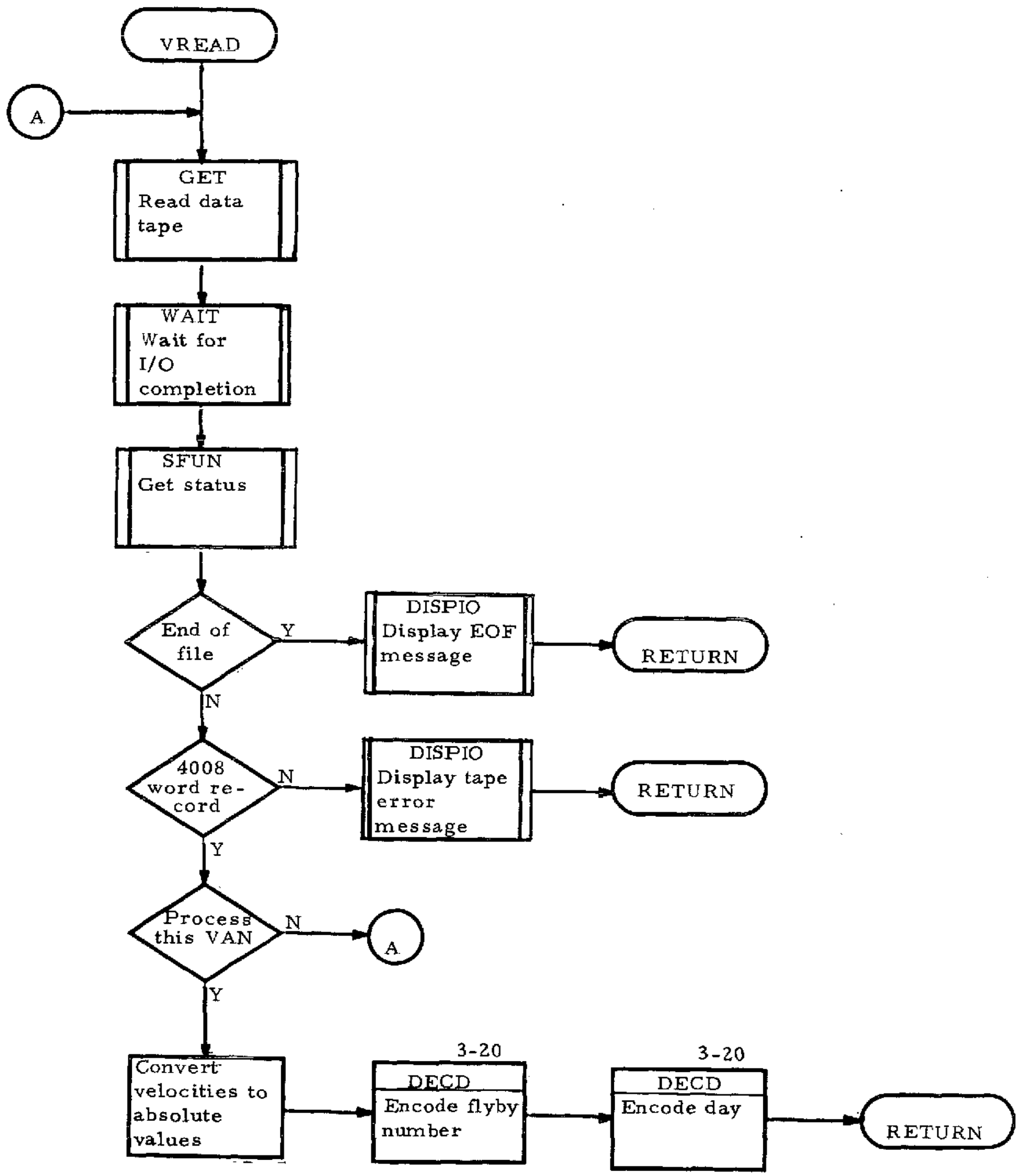

VREAD
A
GET
Read data tape
WAIT
Wait for I/O completion
SFUN
Get status
End of file
Y
DISPIO
Display EOF message
RETURN
N
4008 word re-cord
N
DISPIO
Display tape error message
RETURN
Y
Process this VAN
N
A
Y
Convert velocities to absolute values
3-20
DECD
Encode flyby number
3-20
DECD
Encode day
RETURN

Figure 3-19

External References

GET, WAIT, SFUN, DISPIO, and DECD.

3.8 Encode Integers

Name

DECD

Calling Sequence

CALL DECD (KK, KKK, KP)

Where:

- o KK is the number to be encoded.
- o KKK is the number of bytes output.
- o KP is the address that contains the encoded number.

Description of Function

Subroutine DECD (Figure 3-20) encodes an integer number. The alphanumeric results are stored left-justified. The maximum number of bytes output is five. If the encoded number is greater than five bytes, only the right most bytes are output. Signs are also output and count as one of the five bytes.

External References

None.

3.9 Get Address

Name

SETAD

Calling Sequence

CALL SETAD (I, V)

Where:

- o I is the return variable containing the address of V.

ENCODE INTEGERS

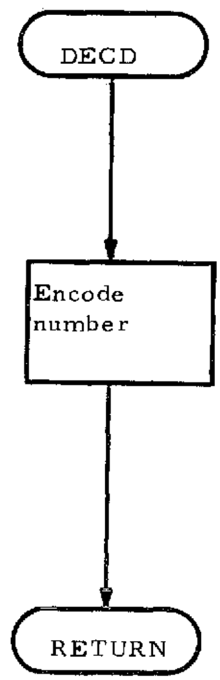


Figure 3-20

- o V is an integer or real variable but must begin on an odd word variable.

Description of Function

SETAD (Figure 3-21) gets the address of a variable.

External References

None.

3.10 Set Plane Type

Name

PTYP

Calling Sequence

CALL PTYP (N)

Where N is a six element array defined as follows:

- o N(1) contains the number of the display that was on the screen when PTYP was called.
- o N(2) contains the option number that was selected from the display.
- o N(3) contains the number of bytes in a compose field.
- o N(4) indicates the type of input field.
- o N(5) and N(6) are input field values.

Description of Function

PTYP (Figure 3-22) determines the name of the aircraft from input data, sets the default values for aircraft dependent parameters, brings up the appropriate aircraft dependent parameters display and fills in the day, flyby number, aircraft type, and default values on this display. In post-analysis, PTYP can also search the data tape to find a desired plane type.

External References

DISPIO, TIME, VREAD, and SFUN.

GET ADDRESS

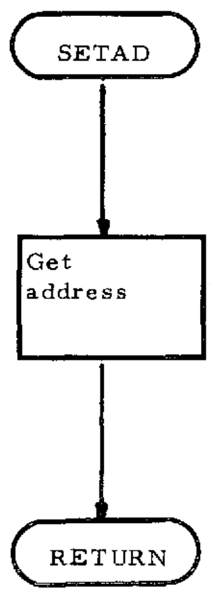


Figure 3-21

SET PLANE TYPE

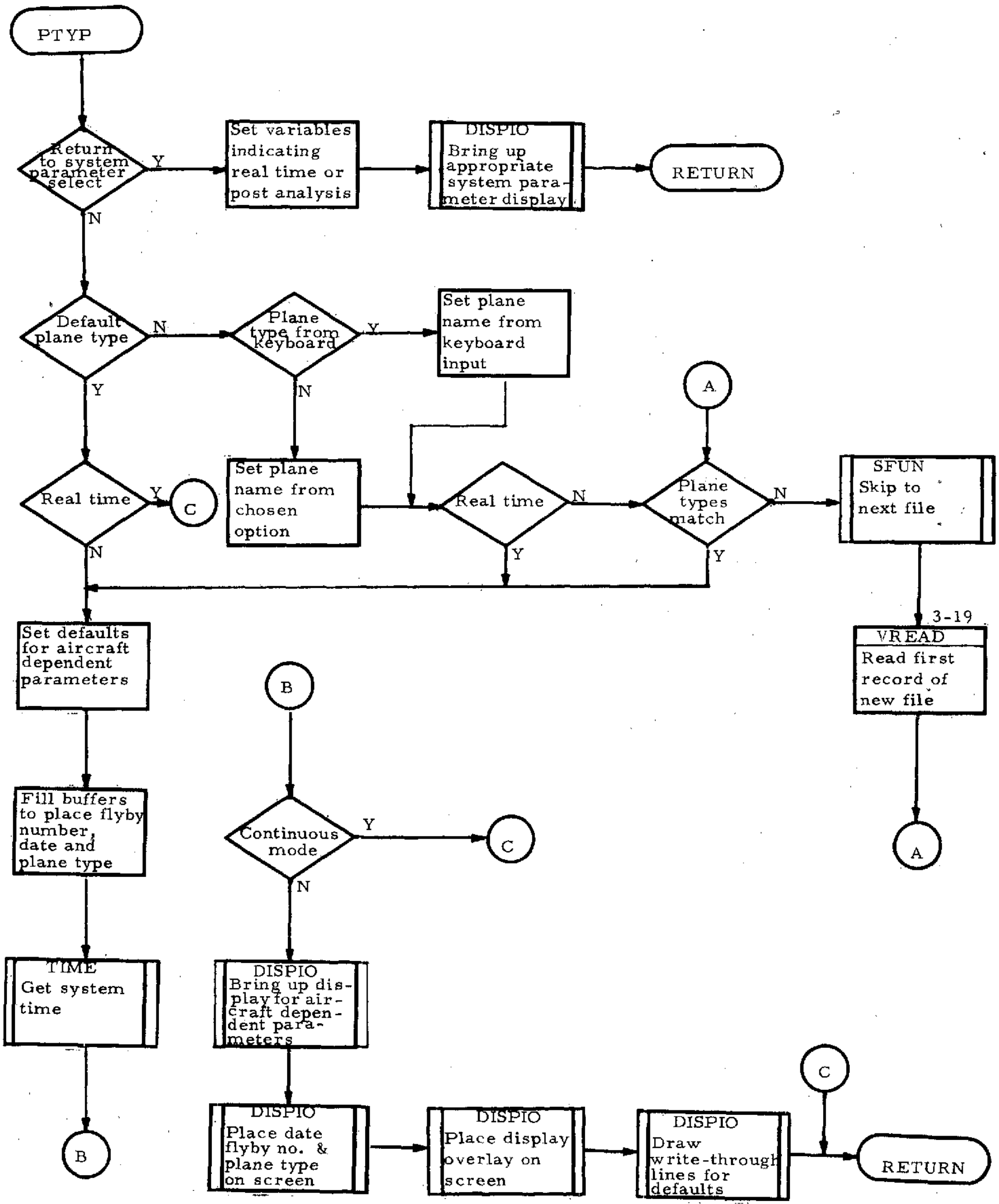


Figure 3-22

3.11 Calculate Velocity

Name

GETVEL

Calling Sequence

CALL GETVEL (IV, VEL)

Where:

- IV is an integer word that contains the peak velocity in bits 8-14 and the maximum velocity in bits 0-6 in counts.
- VEL is a real variable that contains the peak velocity in feet/second.

Description of Function

Subroutine GETVEL (Figure 3-23) extracts the peak velocity in counts from an integer word that contains both the peak velocity and the maximum velocity for a data point. It then converts the integer velocity in counts to a floating point velocity in feet/second according to Table 3-1.

External References

SUBBIT and FLOAT.

3.12 Find Vortex Center

Name

CENTRD

Calling Sequence

CALL CENTRD

Description of Function

CENTRD (Figure 3-24) processes raw data in an attempt to locate vortex centers. The data is checked to see if it possesses the minimum number of points that are required to define a vortex. The minimum number of points to locate a vortex center is never less than two and will be two for the first vortex of a data frame. For the second vortex the minimum is defined as C% of the number of points contained in the first vortex center. If there

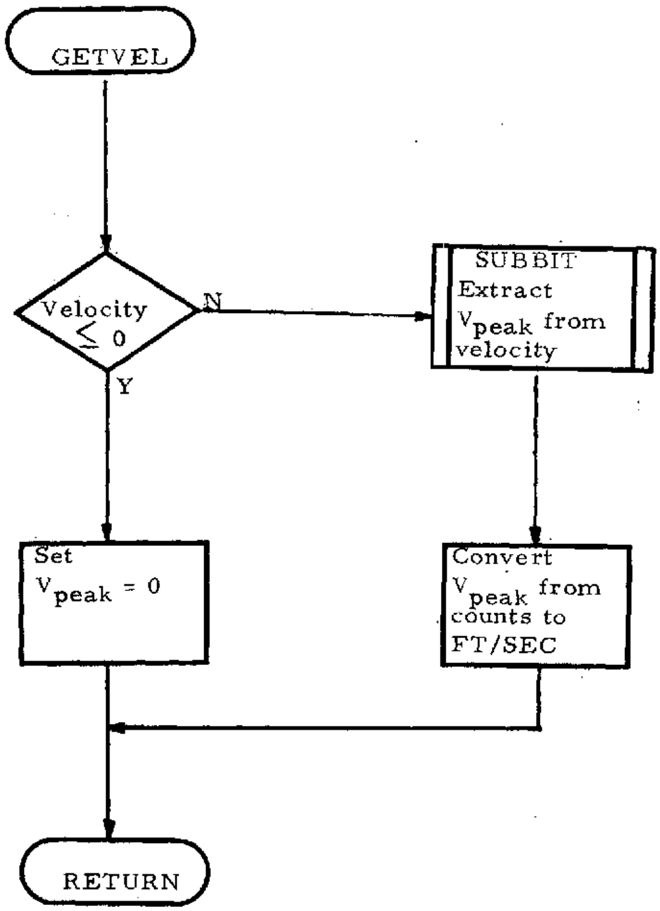

GETVEL
Velocity ≤ 0
N
Y
SUBBIT
Extract V_{peak} from velocity
Set $V_{peak} = 0$
Convert V_{peak} from counts to FT/SEC
RETURN

Figure 3-23

COUNT TO VELOCITY CONVERSION

COUNT	VELOCITY IN FEET PER SECOND
≤ 69	1.8 * count
$70 \leq 75$	1.8 * 69 + (count-69) * 3.6
> 75	1.8 * 69 + 6 * 3.6 + (count-75) * 7.2

Table 3-1

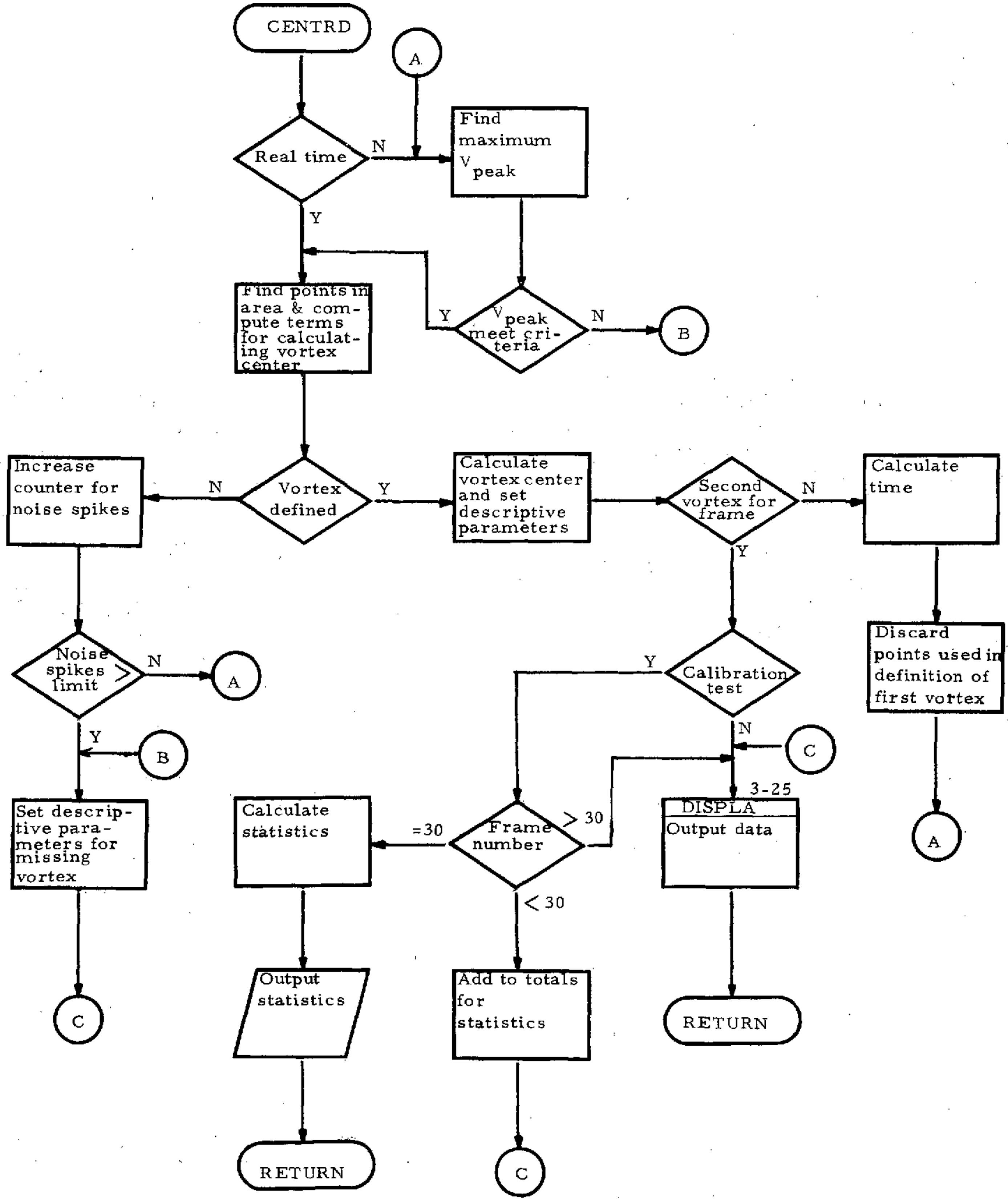

CENTRD
A
Real time
N
Y
Find maximum V_{peak}
Find points in area & compute terms for calculating vortex center
V_{peak} meet criteria
Y
N
B
Increase counter for noise spikes
N
Vortex defined
Y
Calculate vortex center and set descriptive parameters
Second vortex for frame
N
Calculate time
Y
Noise spikes > limit
N
A
Y
Calibration test
Discard points used in definition of first vortex
Y
B
N
C
3-25
DISPLA
Output data
Set descriptive parameters for missing vortex
Calculate statistics
=30
Frame number
> 30
< 30
A
Output statistics
Add to totals for statistics
RETURN
C
RETURN
C

Figure 3-24

are insufficient points, the description parameters that are used in tabular data output for the vortices are set to indicate that no vortex center was found. The assigned values are:

- o NCORPT(1) = NUMPTS
- o NCORPT(2) = 0
- o PKVEL(1) = maximum peak velocity
- o NOISE(1) = 0
- o The remaining descriptive parameters are not set but will be set equal to a blank in subroutine DISPLA.

If there are sufficient points, the point with the maximum peak velocity is found, and its velocity checked to see that it has a minimum value of D% of the maximum peak velocity of the last vortex center found (it is assumed that for the first frame that the previous velocity was zero). If the velocity is less than the minimum, the descriptive parameters are again set as shown above to indicate that no vortex center was found.

If the maximum peak velocity equals or exceeds the minimum, the points in a correlation region defined as all points within a radius R of the point with maximum velocity are located. If there are a sufficient number of points to define a vortex and if B% of the points have a minimum velocity defined as A% of the maximum, then a vortex center is determined using the following equation (see Table 3-2):

$$X = \frac{\sum_{i=1}^{k} I_i \cdot Vpeak_i \cdot X_i}{\sum_{i=1}^{k} I_i \cdot Vpeak_i} \quad \text{and} \quad Y = \frac{\sum_{i=1}^{k} I_i \cdot Vpeak_i \cdot Y_i}{\sum_{i=1}^{k} I_i \cdot Vpeak_i}$$

If the above criterion is not met, the point is rejected as a noise spike, the noise spike count is increased by one, and if the noise spike limit has not been exceeded, the process is repeated beginning with a search for the point possessing the maximum peak velocity. If the noise spike limit is exceeded, the descriptive parameters are set as shown above to indicate that no vortex center was located except NOISE(1) will be equal to the number of noise spikes and NCORPT(1) will be equal to the number of points in the correlation region. Then subroutine DISPLA will be called.

When the first vortex center is found, time is calculated, NOISES(1) is set equal to the noise spikes, NCORPT(1) is set equal to KOUNT, PKVEL(1) is set equal to the maximum peak velocity in the vortex center, all of the points used in determining the location of this vortex center are rejected, and

DEFINITION OF TERMS

SYMBOL	MNEMONIC	DEFINITION	UNITS
<u>Input from LDV's</u>			
I	INTENS (bits 7-14)	intensity	Counts
Vpeak	IVEL (bits 8-14)	peak velocity	Counts
X	IX	horizontal coordinate of point wrt LDV	Counts
Y	IY	vertical coordinate of point wrt LDV	Counts
N	NUMPTS	number of data points	None
<u>Variable Input</u>			
A	VELTOL	A defines the minimum peak velocity that B% of the points in a correlation region must possess by requiring them to have A% of the maximum peak velocity.	None
B	PTTOL	B defines the percent of points in a correlation region that must possess a minimum velocity defined as A% the maximum peak velocity.	None
C	VORTOL	Defines the minimum number of points needed to locate a second vortex center in a frame of data since this number is C% of the number of points which defined the first vortex center.	None
D	FRVTOL	The maximum peak velocity for the first vortex center defined in a data frame must be D% of the peak velocity from the last defined vortex center.	None
R	IRADI	Radius of correlation volume	feet
NS	NOISE	The maximum peak velocity for the definition of the second vortex must be separated from the maximum peak velocity of the first vortex by NS * R.	None

Table 3-2

DEFINITION OF TERMS
(continued)

SYMBOL	MNEMONIC	DEFINITION	UNITS
Variables Calculated in CENTRD			
K	KOUNT	Number of points in a correlation region.	None
Descriptive Parameters Determined in CENTRD			
X	XCG	A two element array containing the horizontal distances between the vortex centers and the LDV.	feet
Y	YCG	A two element array containing the vertical distances between the vortex centers and the LDV	feet
n	NCORPT	A two element array containing the number of points in the correlation regions of the vortices.	None
(Vpeak)max	PKVEL	A two element array containing the maximum peak velocity for each vortex.	ft/sec
Θmin	ELANG1	Minimum angle at which data point was found.	deg.
Θmax	ELANG2	Maximum angle at which a data point was found.	deg.
NVORTX	NVORTX	Number of vortices found in a frame of data.	None
t	RTIME	Time at which the vortex center was detected relative to time at which the first data point was detected.	sec.
NOISE	NOISES	A two element array containing the number of noise spikes that were encountered while locating a vortex center. Maximum number allowed is 5.	

Table 3-2
(continued)

a search using the same techniques is used to locate a second vortex center. However, when descriptive parameters are set they will have an index of 2 and there is no velocity check against the last vortex center found. When either the second vortex center has been located or it has been determined that a second vortex center cannot be defined, subroutine DISPLA is called to output the data.

External References

DISPLA, GETVEL, and SUBBIT.

3.13 Display Output Data

Name

DISPLA

Calling Sequence

CALL DISPLA

Description of Function

Subroutine DISPLA (Figure 3-25) transforms vortex locations from LDV referenced coordinate systems to the center of runway coordinate system, encodes description parameters for output, sets the display controller buffers with the desired output format, and calls the display controller to output data. For a detailed description of output choices and output see Section 6 of this manual.

External References

DISPIO.

3.14 Terminate Program

Name

TERM

Calling Sequence

CALL TERM (N)

Where: N is a dummy argument.

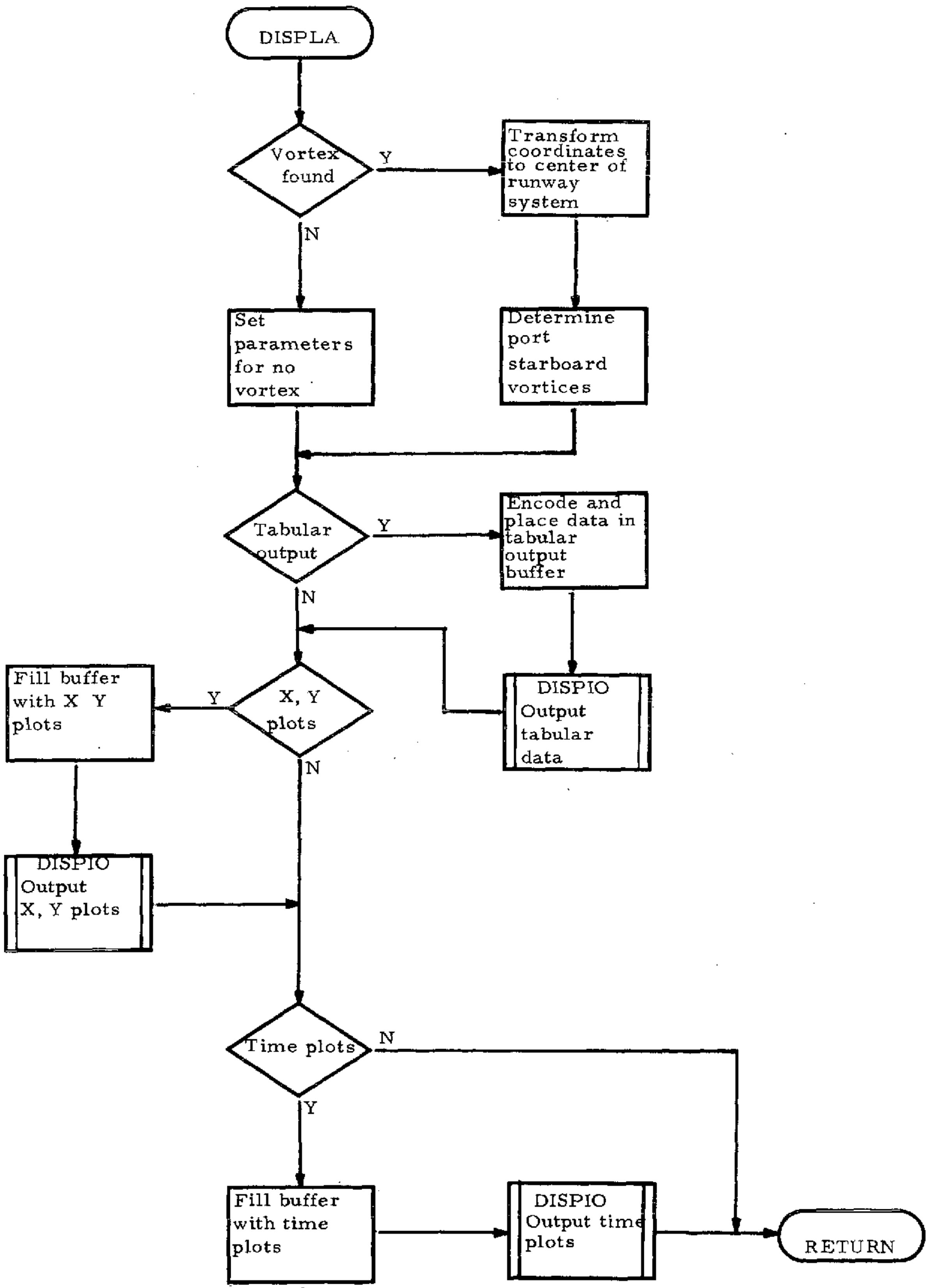

DISPLA
Vortex found
Y
Transform coordinates to center of runway system
N
Set parameters for no vortex
Determine port starboard vortices
Tabular output
Y
Encode and place data in tabular output buffer
N
DISPIO Output tabular data
Fill buffer with X Y plots
Y
X, Y plots
N
DISPIO Output X, Y plots
Time plots
N
Y
Fill buffer with time plots
DISPIO Output time plots
RETURN

Figure 3-25

Description of Function

Subroutine TERM (Figure 3-26) prepares the data tape for program termination and returns control to the DOS Monitor. When the program is operating in real time and recording data on magnetic tape, TERM writes two consecutive end of files at the end of data. The routine then rewinds the tape for both real time and post-analysis and returns control to the DOS Monitor.

External References

SFUN, WAIT, CLOSE, and DISPIO.

3.15 Scatter Plot Generation

Name

SCAT

Calling Sequence

CALL SCAT

Description of Function

Subroutine SCAT (Figure 3-27) plots the raw data points and the vortex centers in an X-Y coordinate system for each data frame in a flyby. The character which represents each point is determined by the magnitude of the velocity for that point. The points possessing the ten highest velocities are represented by the corresponding first ten letters of the alphabet. Subsequent points are represented according to the following table:

SYMBOL	VELOCITY IN FT. /SEC.
0	20-30
1	30-40
2	40-50
3	50-60
4	60-70
5	70-80
6	80-90
7	90-100
8	100-110
9	110-120

External References

CENTRD, DISPIO, GETVEL, DECD, SETAD, VREAD, DBUG, and OPEN.

TERMINATE PROGRAM

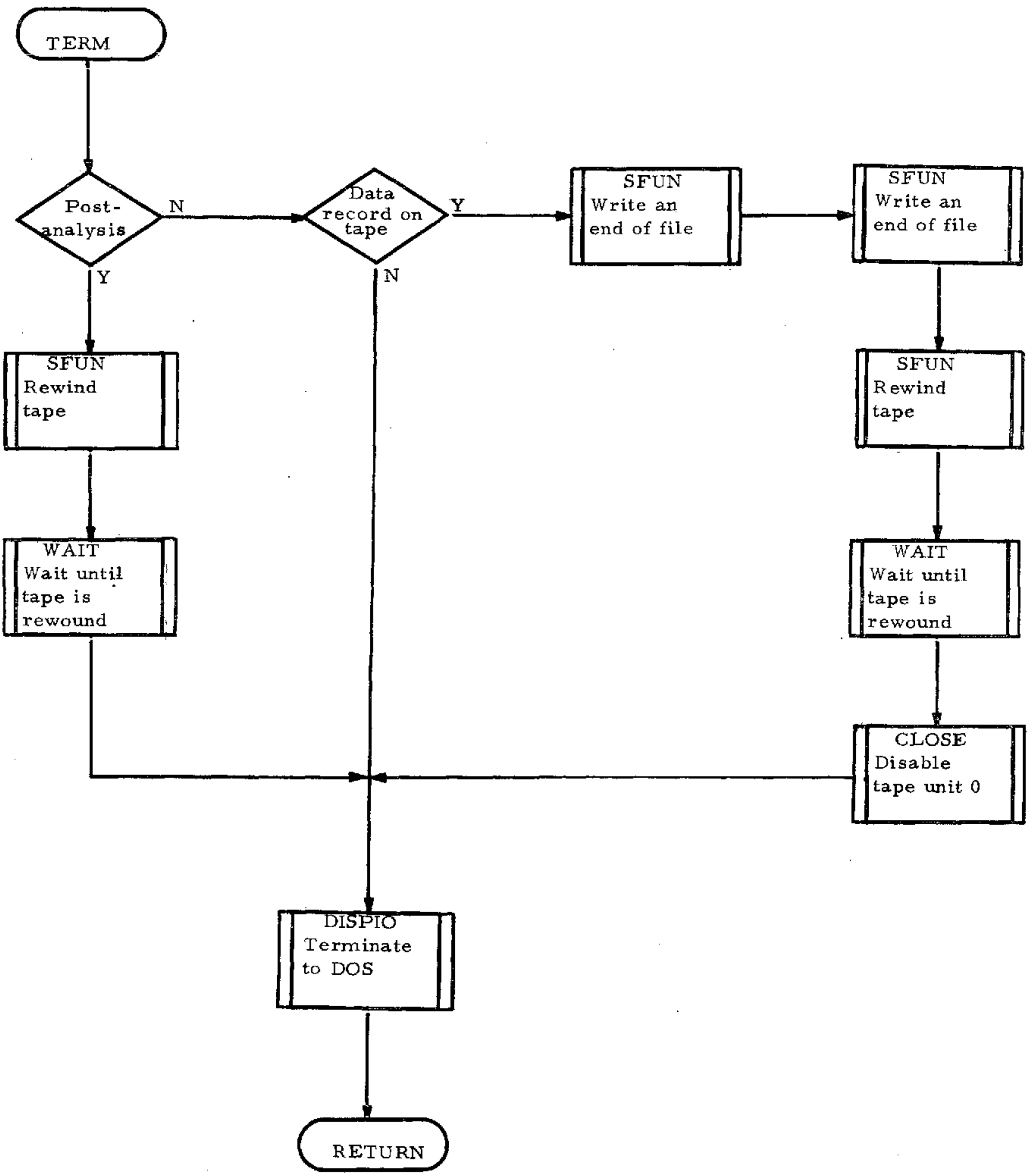


Figure 3-26

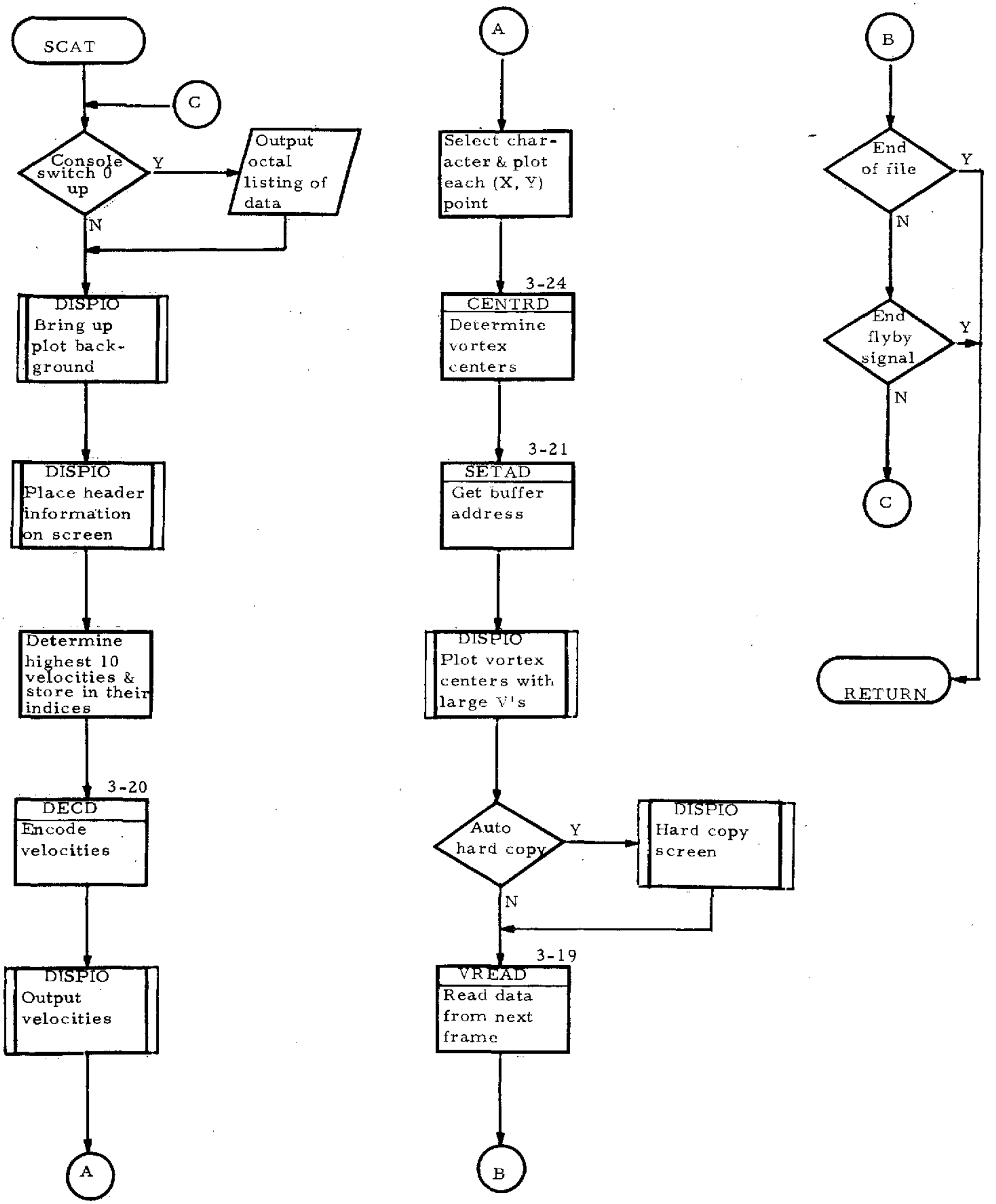

SCAT
C
Console switch 0 up
Y
N
Output octal listing of data
DISPIO
Bring up plot back-ground
DISPIO
Place header information on screen
Determine highest 10 velocities & store in their indices
3-20
DECD
Encode velocities
DISPIO
Output velocities
A
A
Select char-acter & plot each (X, Y) point
3-24
CENTRD
Determine vortex centers
3-21
SETAD
Get buffer address
DISPIO
Plot vortex centers with large V's
Auto hard copy
Y
N
DISPIO
Hard copy screen
3-19
VREAD
Read data from next frame
B
B
End of file
Y
N
End flyby signal
Y
N
C
RETURN

Figure 3-27

4. DISPLAY CONTROLLER

The Display Controller coordinates all communication between the operator at the display terminal and the application program (Vortex) executing in the PDP-11 computer. In particular, the Display Controller performs the following functions:

1. processes data tablet inputs from the operator,
2. processes keyboard inputs from the operator,
3. processes information outputs from the application program (Vortex),
4. processes display control information from the Display Library, and
5. performs all input and output to the terminal.

Basically, the Display Controller is organized as two processors to perform the primary functions listed above. The two processors are:

- User Input Processor (1 and 2)
- Application Program Request Processor (3)

In addition, there are routines that are essential for the proper execution of these processors but are not a part of either one. They perform all accessing of the Display Library information and the input/output processing. They will be referred to in this section as:

- Common Utility routines (4)
- Input/Output Interrupt Processing (5)

Figure 4-1 illustrates the structure of the Display Controller and how its processors interface with external devices and the application program.

Descriptions and flowcharts of the Display Controller, as categorized above, follow.

4.1 User Input Processor

The User Input Processor (whose component block diagram is depicted in Figure 4-2) processes both data tablet inputs and keyboard inputs from the operator at the display terminal. In response to data tablet inputs, it outputs a graphic cursor that tracks the position of the data tablet cursor on the tablet. This processor also passes information to the application program regarding points selected by the operator through depression of the data tablet cursor.

DISPLAY CONTROLLER

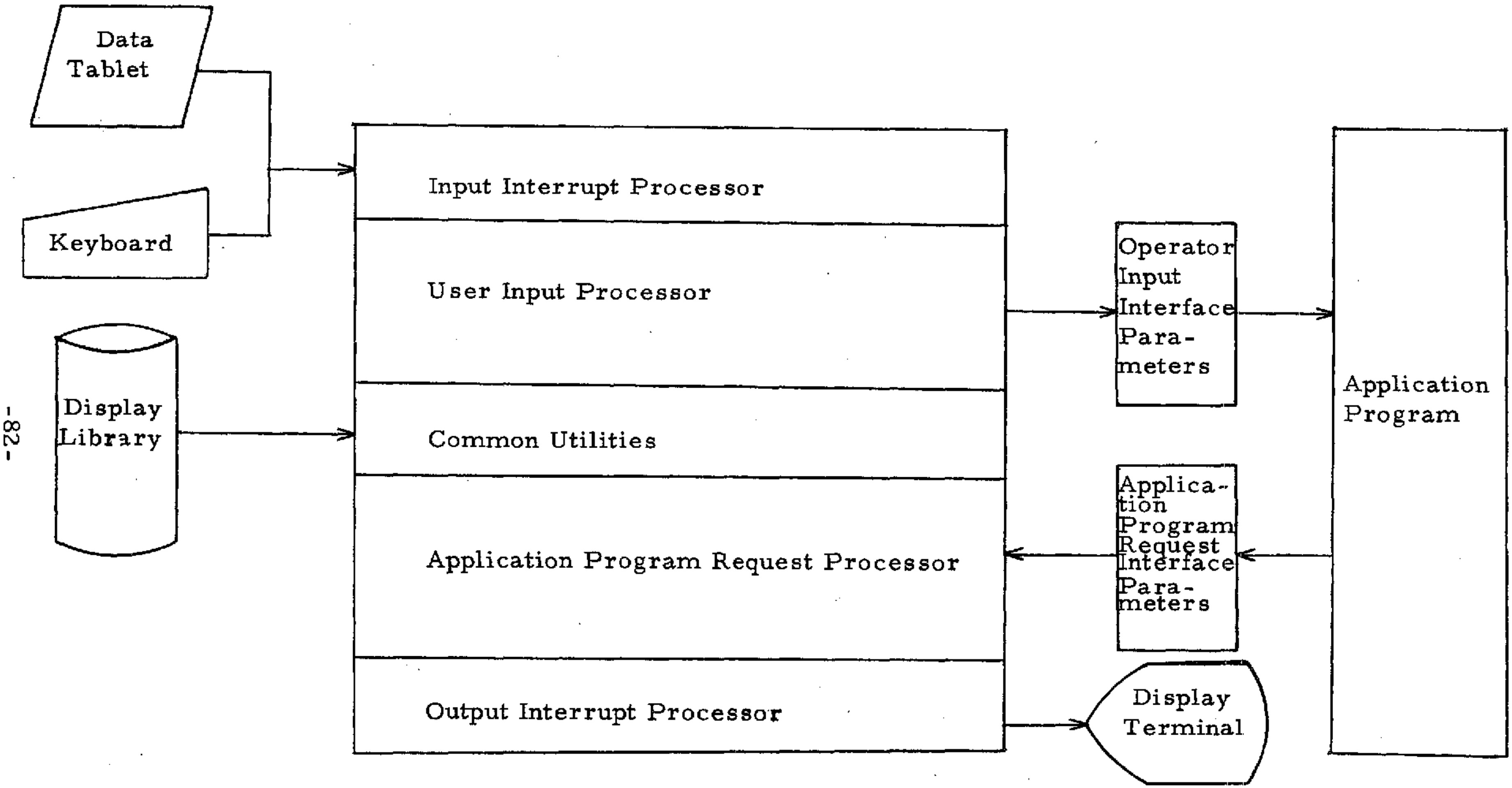


Figure 4-1

USER INPUT PROCESSOR COMPONENTS

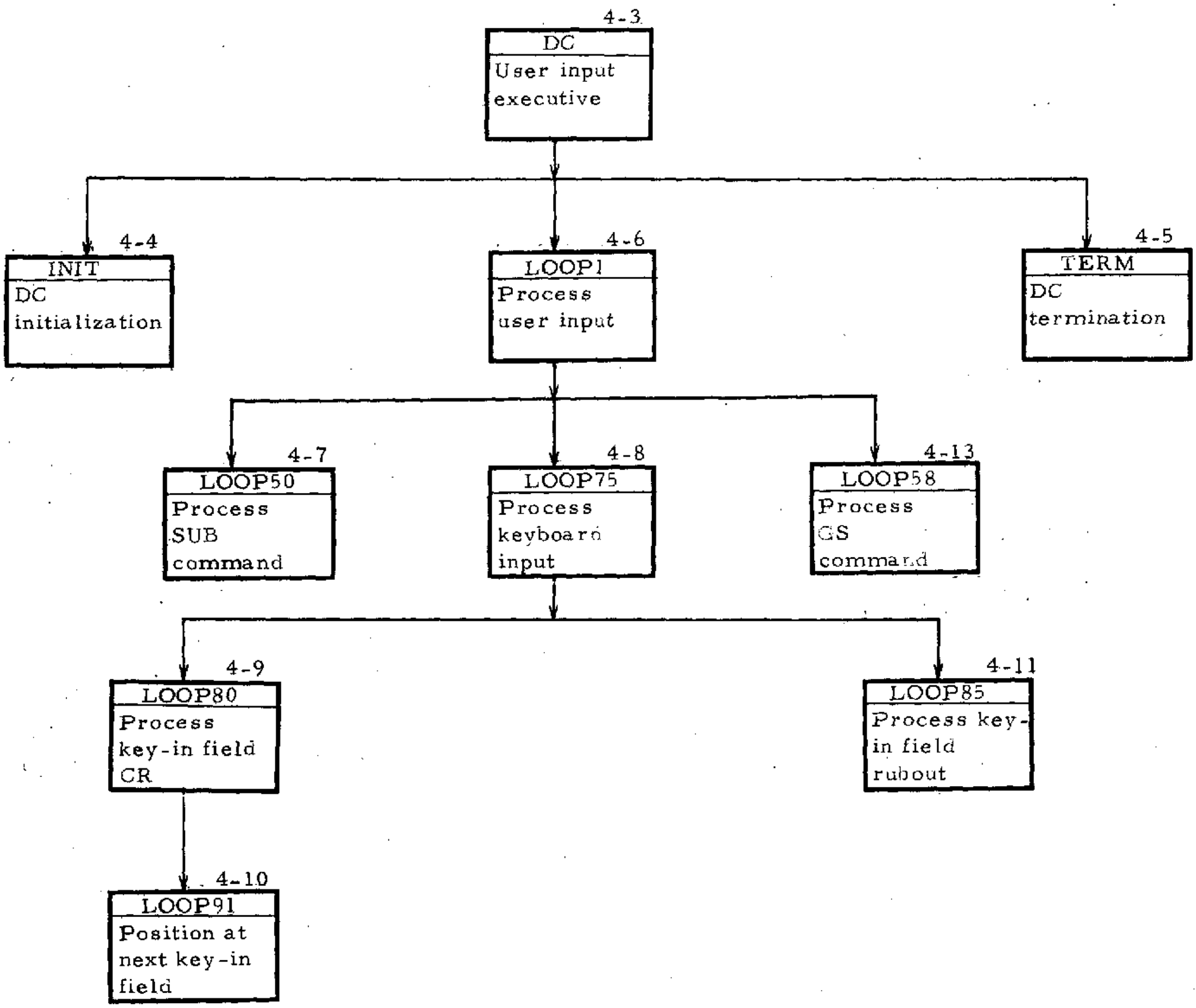


Figure 4-2

Processing keyboard inputs involves both echoing the keyed-in data back to the operator at the terminal and passing the data to the application program for further processing.

Various application program requests are referenced in this section. The processing of these requests is discussed in Section 4.2.

User Input Executive

The User Input Executive (DC (Figures 4-3 through 4-5)) initiates processing. This routine performs initialization of the DOS disk driver, data tablet input, and certain hardware addresses needed by the controller. It calls ATVWS and ONCE to perform various initialization functions for the application program. DC then brings up the first display on the screen and initiates the input search loop (LOOP1) which continues to process for the duration of execution.

When execution has been terminated, DC then releases the DOS disk driver, terminates data tablet input, and resets those hardware addresses previously initialized by it.

Process User Input

The Process User Input routine (LOOP1 (Figure 4-6)) repeatedly checks for user input from the data tablet and the keyboard which if present has been extracted from the hardware input register and stored in a table for this routine by the Input Interrupt Processor. LOOP1 searches this table for three specific types of input:

- o Data tablet SUB command coordinates (data tablet cursor is in proximity of data tablet)
- o Keyboard character
- o Data tablet GS command (point/option selected via data tablet cursor by operator)

If input is present, it is processed by the appropriate routine. LOOP20 is then executed to refresh the screen(s). LOOP1 then determines if the application program has requested termination through the Reset Request to the controller. As long as termination has not been requested, LOOP1 repeats its search for input.

Process SUB Command

Process SUB Command routine (LOOP50 (Figure 4-7)) processes the SUB command input received from the data tablet whenever the data tablet cursor is in proximity of the tablet. This processing consists of using the coordinates that are transmitted with the SUB command to activate and build an output string that displays the graphic cursor on the screen to indicate the current position of

USER INPUT EXECUTIVE

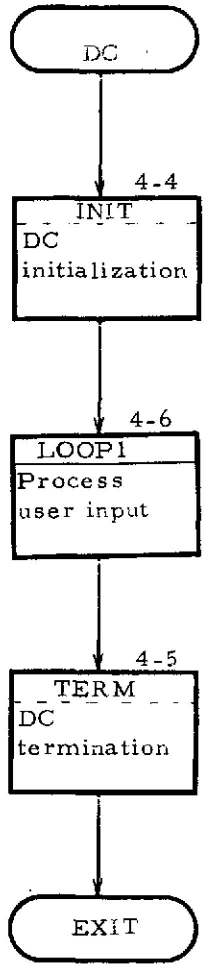


Figure 4-3

DC INITIALIZATION

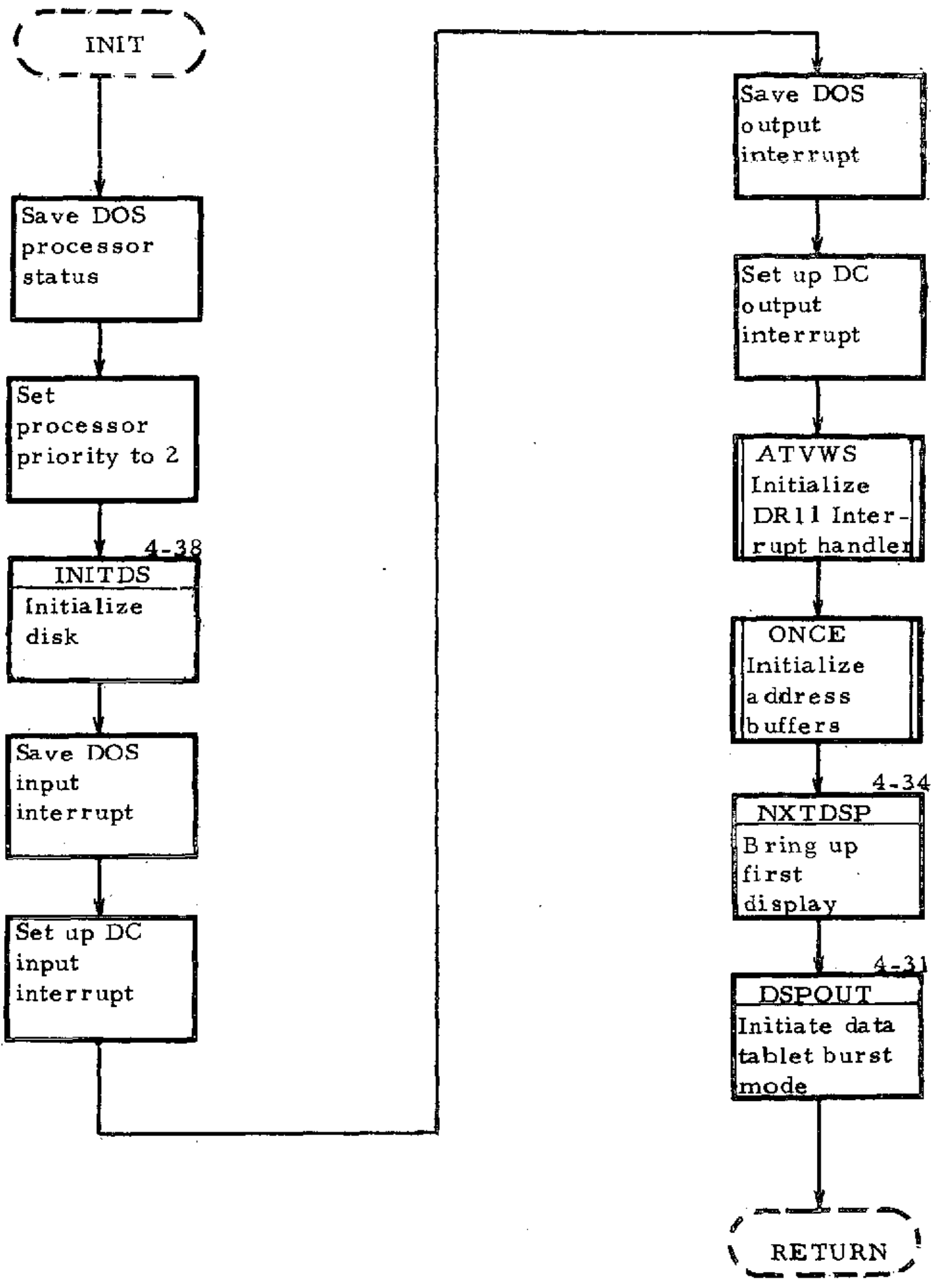


Figure 4-4

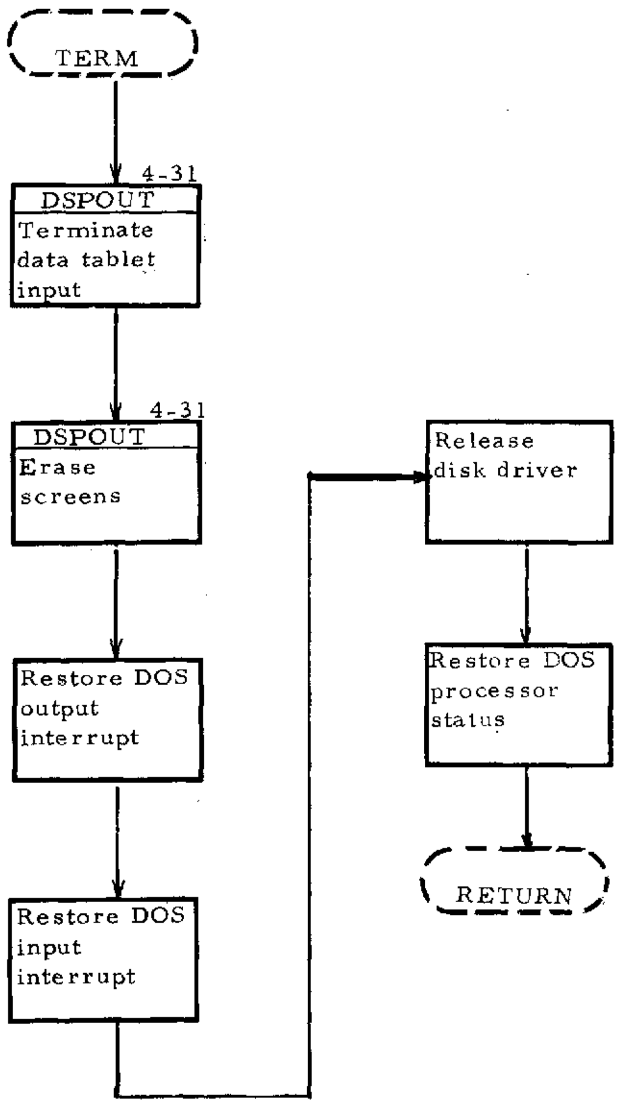

TERM
4-31
DSPOUT
Terminate data tablet input
4-31
DSPOUT
Erase screens
Restore DOS output interrupt
Restore DOS input interrupt
Release disk driver
Restore DOS processor status
RETURN

Figure 4-5

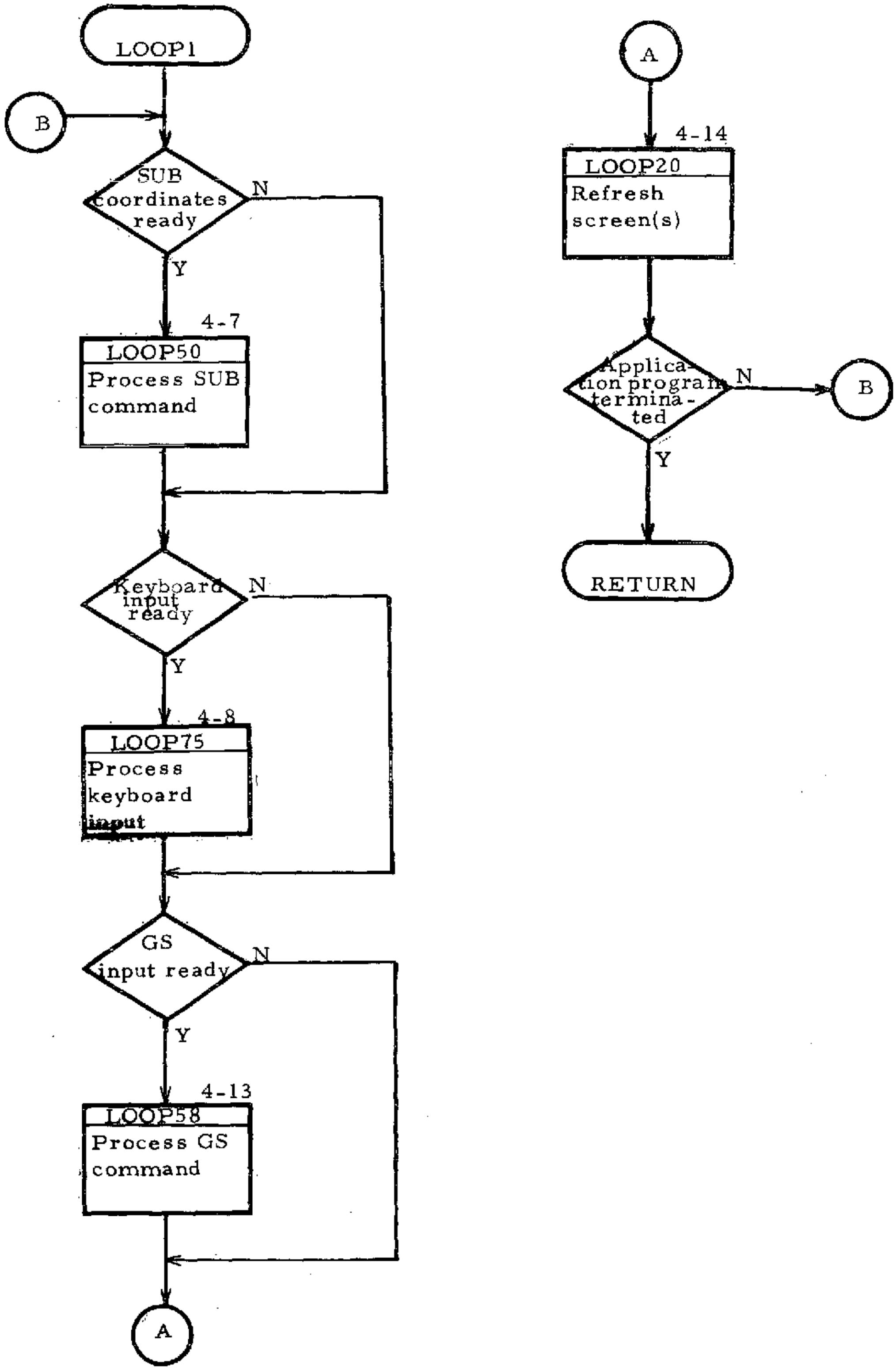

LOOP1
B
SUB coordinates ready
N
Y
4-7
LOOP50
Process SUB command
Keyboard input ready
N
Y
4-8
LOOP75
Process keyboard input
GS input ready
N
Y
4-13
LOOP58
Process GS command
A
A
4-14
LOOP20
Refresh screen(s)
Application program terminated
N
B
Y
RETURN

Figure 4-6

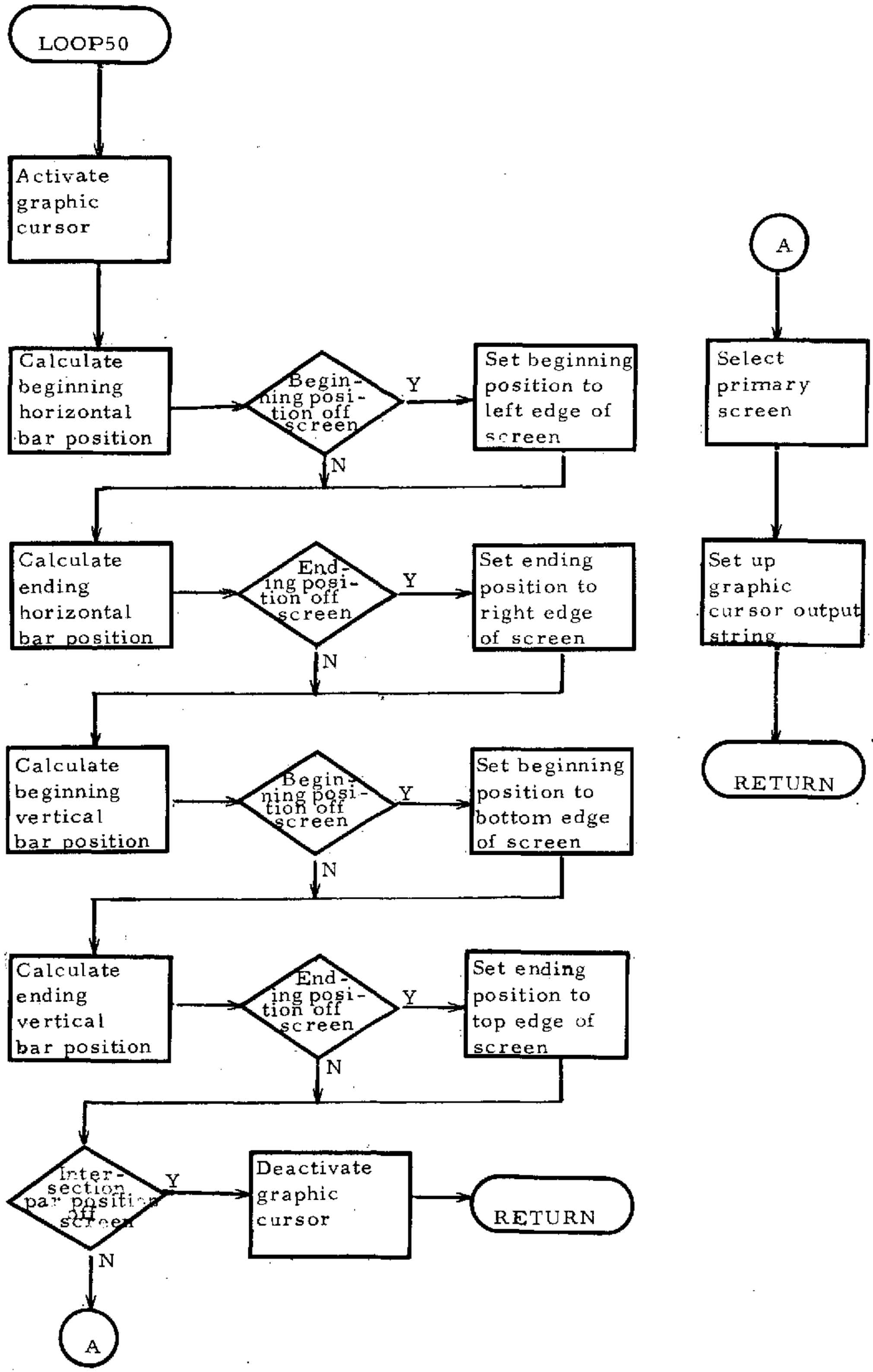

LOOP50
Activate graphic cursor
Calculate beginning horizontal bar position
Beginning position off screen
Y
Set beginning position to left edge of screen
N
Calculate ending horizontal bar position
Ending position off screen
Y
Set ending position to right edge of screen
N
Calculate beginning vertical bar position
Beginning position off screen
Y
Set beginning position to bottom edge of screen
N
Calculate ending vertical bar position
Ending position off screen
Y
Set ending position to top edge of screen
N
Intersection bar position off screen
Y
Deactivate graphic cursor
RETURN
N
A
A
Select primary screen
Set up graphic cursor output string
RETURN

Figure 4-7

the data tablet cursor. The graphic cursor consists of a horizontal bar and a vertical bar intersecting at the point being tracked.

As the coordinates are being converted to the output string format, any part of the generated cursor that exceeds the range of the data tablet (and therefore the display screen) is clipped to prevent wraparound vectors from appearing on the screen. Also, if the data tablet position being tracked is outside the viewing area of the screen but still in range of the tablet (i.e., in the menu area), the graphic cursor output string is deactivated to eliminate unnecessary output to the display.

The graphic cursor is sent only to the primary screen. The primary screen is that screen so designated by the application program through an Auxiliary Screen request. The default is screen 1 (the screen with the keyboard).

Process Keyboard Input

The Process Keyboard Input (LOOP75 (Figures 4-8 through 4-11)) processes all input keyed-in by the operator at the display terminal.

Key-in field data is that data which is input in response to a key-in field designated by "_" characters in the text of the display. The positioning of the alphanumeric cursor always indicates the next available field position for such input.

There are two characters that have special meaning as keyboard input. These characters are the carriage return and the rubout.

The carriage return character indicates end of input and causes the data already keyed-in to be transmitted to the application program's designated program. This routine must have access to the key page of the Display Library for the display currently on the screen. If this 'page' is not currently in core, the LIBINP routine reads it from disk where the library resides.

If there is no keyboard input prior to the carriage return, the alphanumeric cursor is positioned at the next/first key-in field via the information in the key page. If there is prior data, it is passed to the application program's next program (specified in the key page) via ROOTEX for processing.

The input data is set up along with other control information in the format shown in Figure 4-12. Each parameter is described below:

Terminal ID: Identifies the user terminal through which the input was transmitted. This value is significant only for multi-terminal systems.

Next Program: Contains the 4-character name of the program to be executed within the application program to process the operator input.

Current Display: Contains the 4-digit decimal number of the display that is currently being presented to the operator.

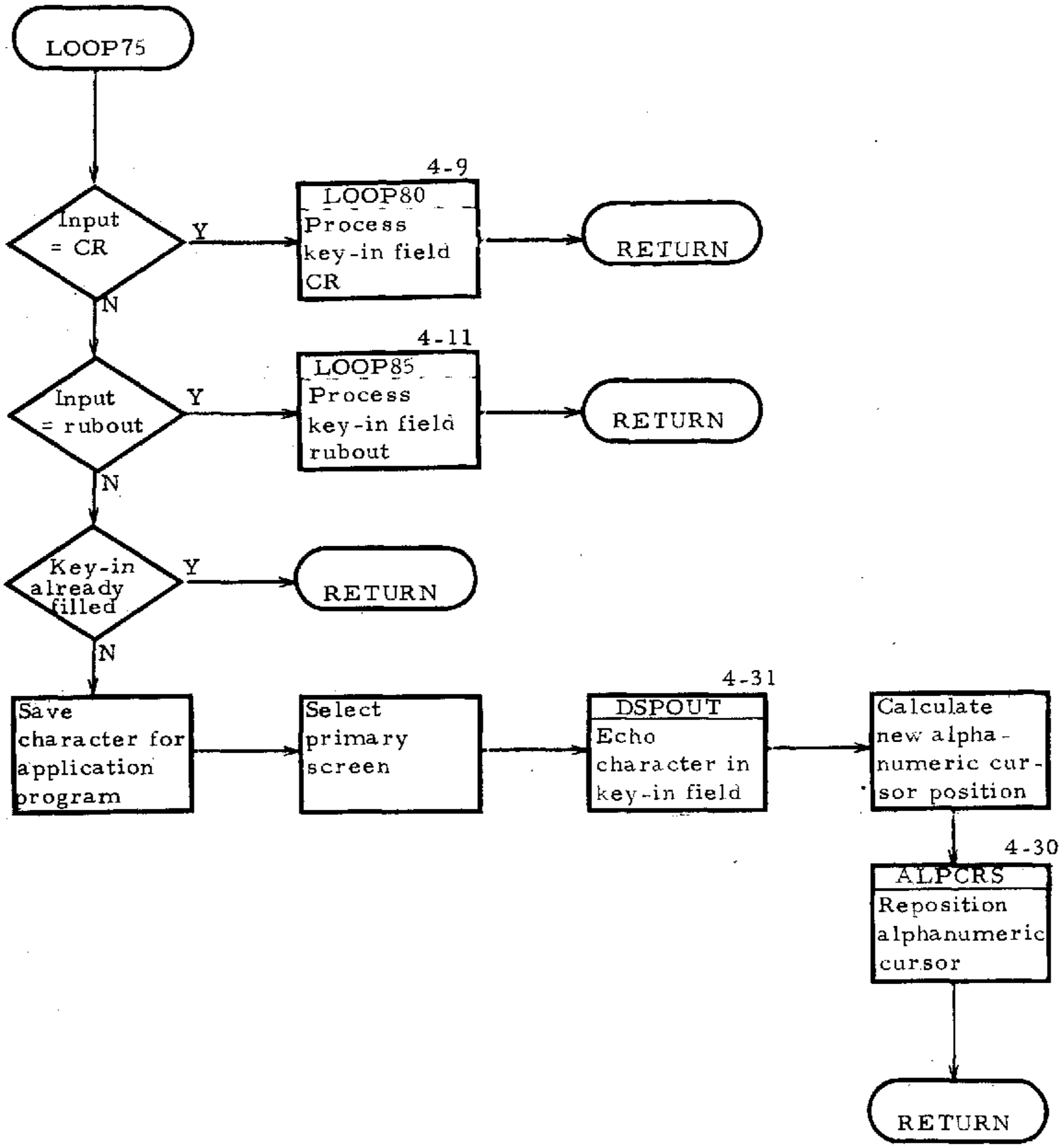

LOOP75
Input = CR
Y
N
4-9
LOOP80
Process key-in field CR
RETURN
4-11
Input = rubout
LOOP85
Process key-in field rubout
RETURN
Key-in already filled
RETURN
Save character for application program
Select primary screen
4-31
DSPOUT
Echo character in key-in field
Calculate new alpha-numeric cursor position
4-30
ALPCRS
Reposition alphanumeric cursor
RETURN

Figure 4-8

PROCESS KEY-IN FIELD CARRIAGE RETURN

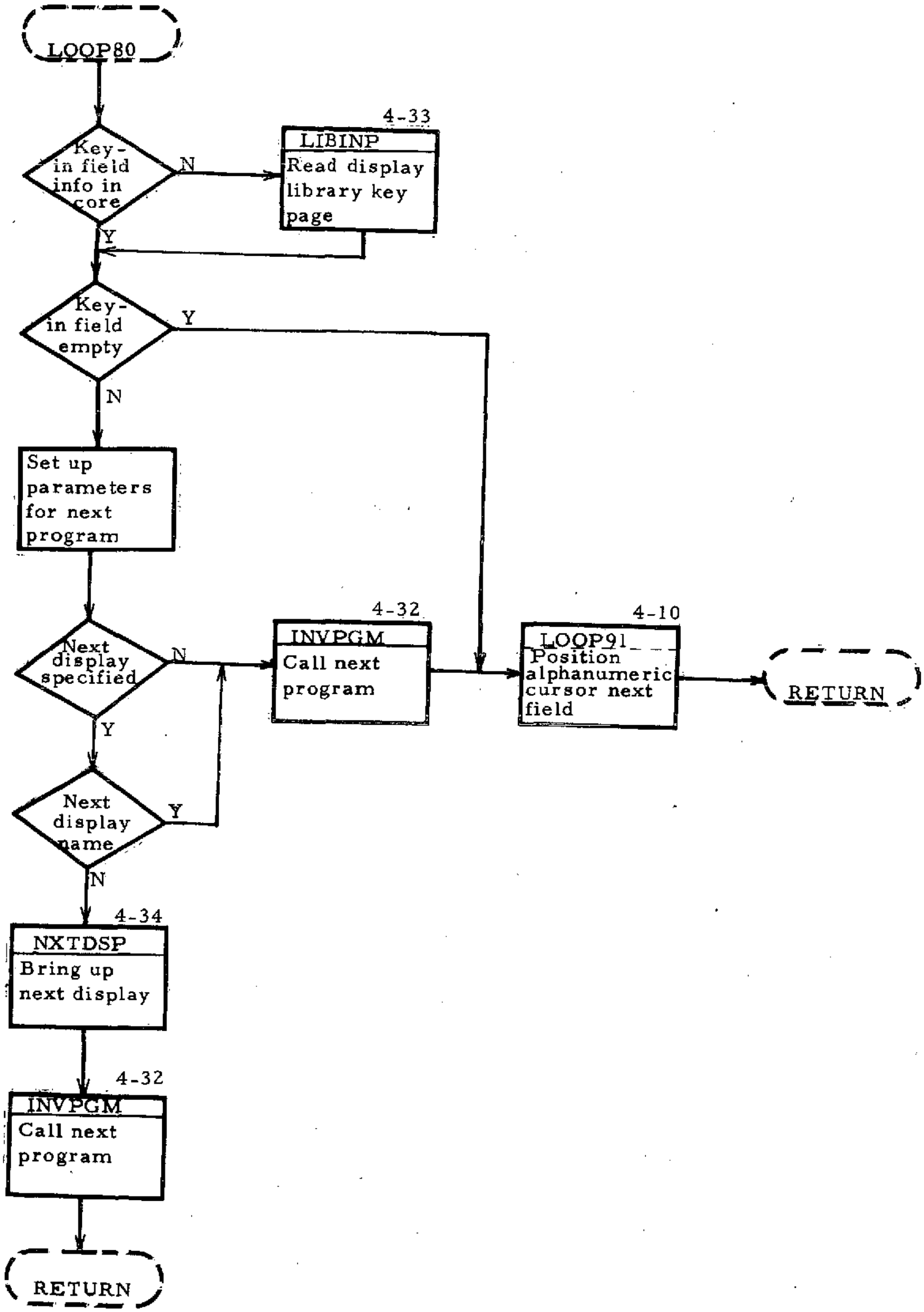


Figure 4-9

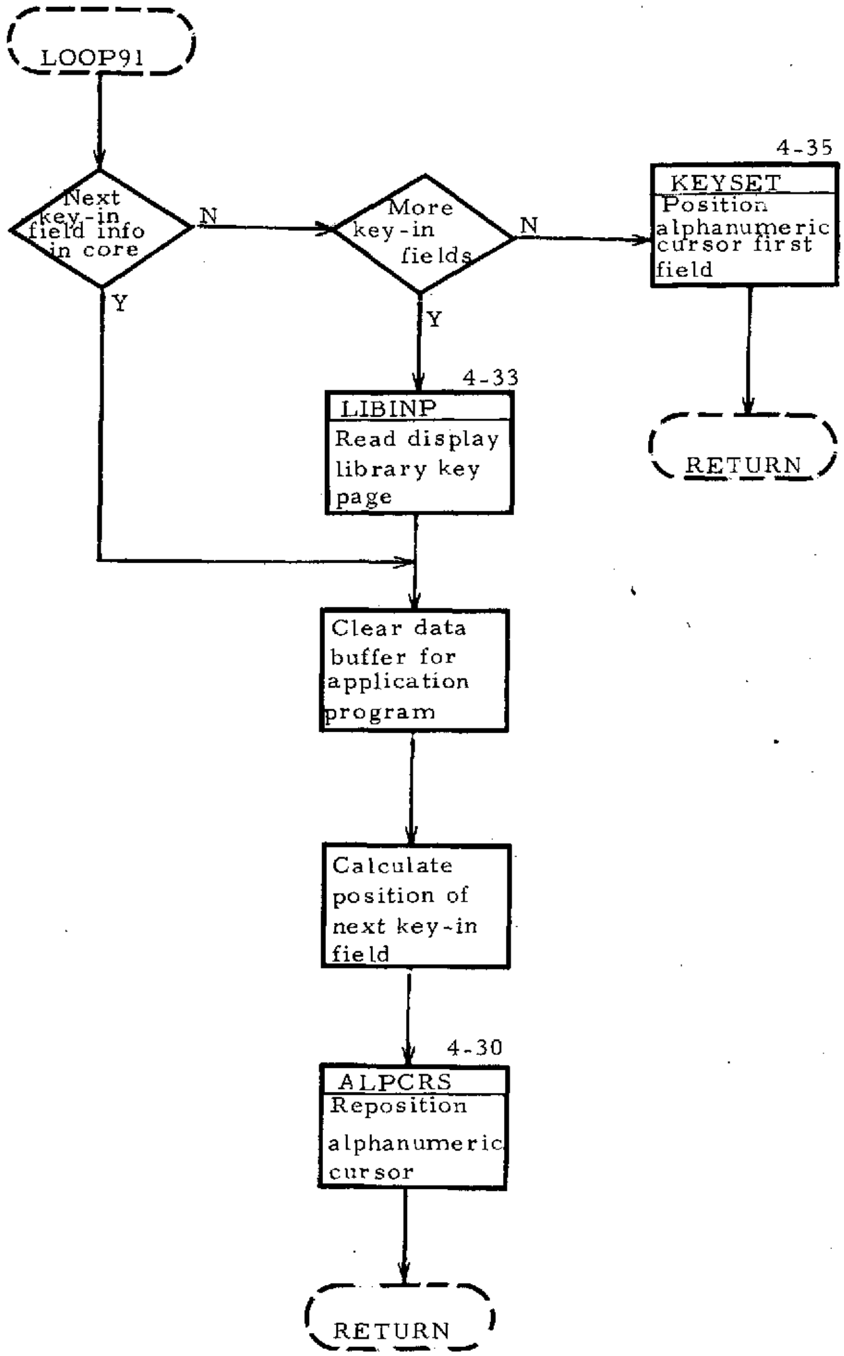

LOOP91
Next key-in field info in core
N
Y
More key-in fields
N
Y
4-35
KEYSET
Position alphanumeric cursor first field
RETURN
4-33
LIBINP
Read display library key page
Clear data buffer for application program
Calculate position of next key-in field
4-30
ALPCRS
Reposition alphanumeric cursor
RETURN

Figure 4-10

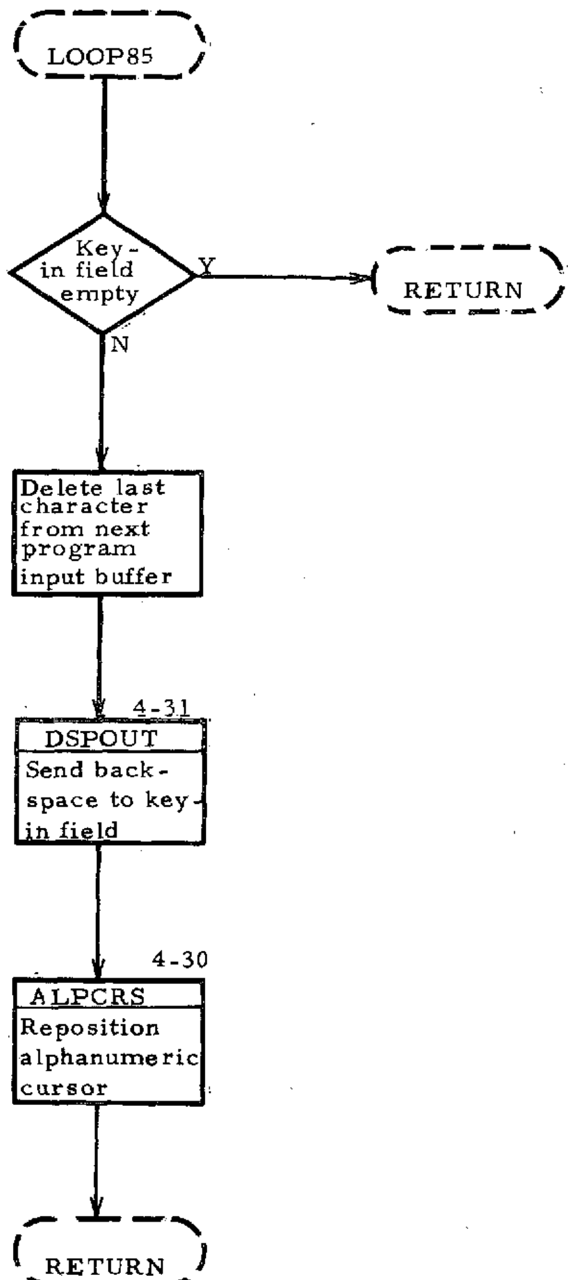

LOOP85
Key-in field empty
Y
RETURN
N
Delete last character from next program input buffer
4-31
DSPOUT
Send back-space to key-in field
4-30
ALPCRS
Reposition alphanumeric cursor
RETURN

Figure 4-11

OPERATOR INPUT INTERFACE PARAMETER LIST

Word 1	Terminal ID
Word 2	Next Program
Word 3	Next Program
Word 4	Current Display
Word 5	Option Number
Word 6	Data Length
Word 7	Data Type
Word 8, etc.	Data Buffer

Figure 4-12

Option Number: Contains the number of the compose field or option selection within the input display associated with this transmission.

Data Length: Contains the number of characters (bytes) of data being sent to the application program.

Data Type: Designates which type of input is being transmitted to the application program.

0 - data tablet option selection

1 - keyboard compose field

2 - data tablet image design point

3 - keyboard image design character

Data Buffer: Contains the input characters (bytes) being transmitted to the application program.

If the key page indicates a next display is associated with the key-in field, NXTDSP routine is executed to bring up the display and then the next program (also indicated in the key page) is called to process the data. If a next display is not specified, the next program is called and the alphanumeric cursor is positioned at the next/first key-in field of the current display.

The rubout character indicates that the character keyed-in previously is to be ignored. If there is no previous data, the rubout is ignored. If there is data, the last character is deleted as input and the alphanumeric cursor is repositioned at the previous character position.

If the keyboard input character is neither the carriage return nor the rubout, it is saved as valid input providing the input field is not already full. The key-in field is limited to the size specified in the key page. The keyboard input character is then echoed on the screen. The character keyed-in to a key-in field is displayed in permanent store mode over the '_' character in the display and the alphanumeric cursor is positioned at the next '_'.

Process GS Command

The Process GS Command routine (LOOP58 (Figure 4-13)) processes the GS command input received from the data tablet whenever a point is selected with the data tablet cursor. The graphic cursor is deactivated and the library pen page is read into core.

The point selected must correspond to a pen option in the pen page for the input to be valid. The pen page is searched until the pen option matching the selected point coordinates is found. If a next program is specified for the selected pen option,

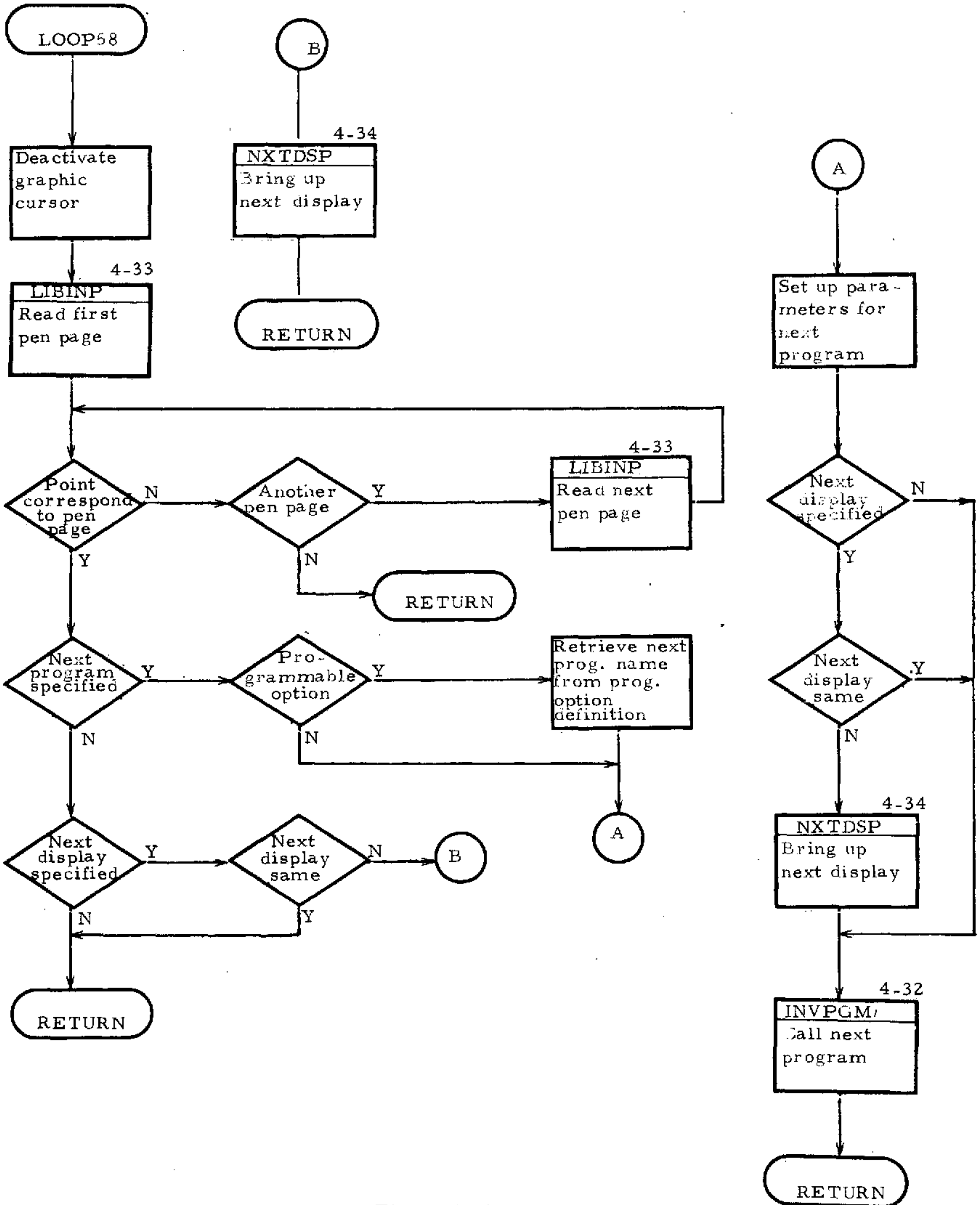

LOOP58
Deactivate graphic cursor
4-33
LIBINP
Read first pen page
Point correspond to pen page
N
Y
Another pen page
Y
N
4-33
LIBINP
Read next pen page
RETURN
Next program specified
Y
N
Pro-grammable option
Y
N
Retrieve next prog. name from prog. option definition
A
Next display specified
Y
N
Next display same
N
Y
B
RETURN
B
4-34
NXTDSP
Bring up next display
RETURN
A
Set up parameters for next program
Next display specified
N
Y
Next display same
Y
N
4-34
NXTDSP
Bring up next display
4-32
INVPGM/
Call next program
RETURN

Figure 4-13

the parameter list is set up (Figure 4-12). If a next display is also specified, it is brought up on the screen by the NXTDSP routine and the next program is executed through the INVPGM routine. If a next program is not specified but a next display is, it is brought up by NXTDSP.

Refresh Screen

The Refresh Screen routine (LOOP20 (Figure 4-14)) initiates all refresh mode output to the screen. This includes:

- graphic cursor in response to SUB command from the data tablet,
- refresh message in response to Refresh Message application program request, and
- keyboard input echo in response to keyboard command input.

4.2 Application Program Request Processor

The Application Program Request Processor (whose block diagram is depicted in Figure 4-15) processes all display I/O requests from the application program. There are 11 types of requests that are handled by this processor. They are:

- tabular output
- new display
- one-line message
- character plot
- vector plot
- erase screen(s)
- refresh message
- hard copy screen(s)
- auxiliary screen
- reset options
- high-speed output

The application program communicates its request to the Display Controller through a standard parameter list as shown in Figure 4-16 with associated buffer formats shown in Figure 4-17.

REFRESH SCREEN

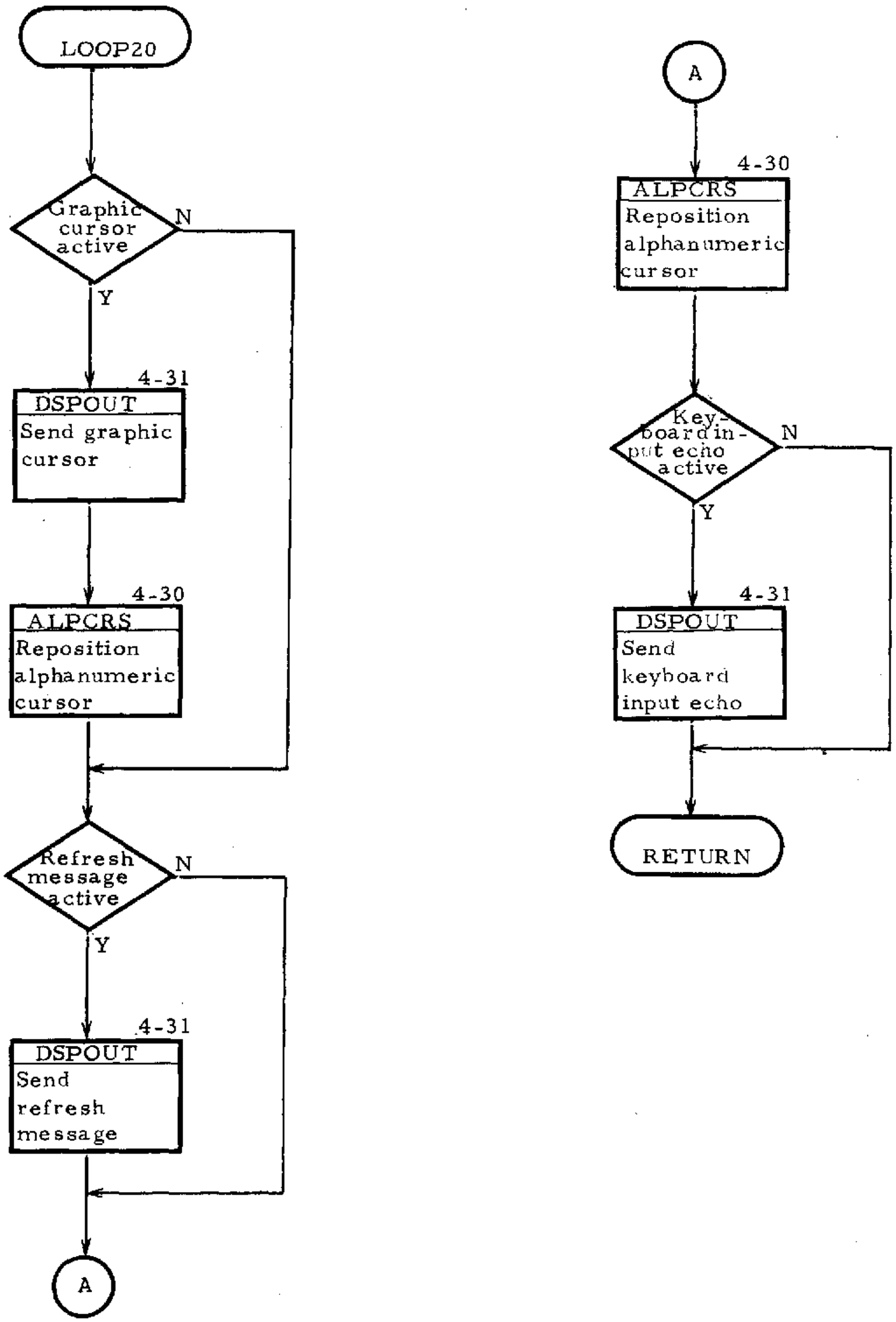


Figure 4-14

APPLICATION PROGRAM REQUEST PROCESSOR COMPONENTS

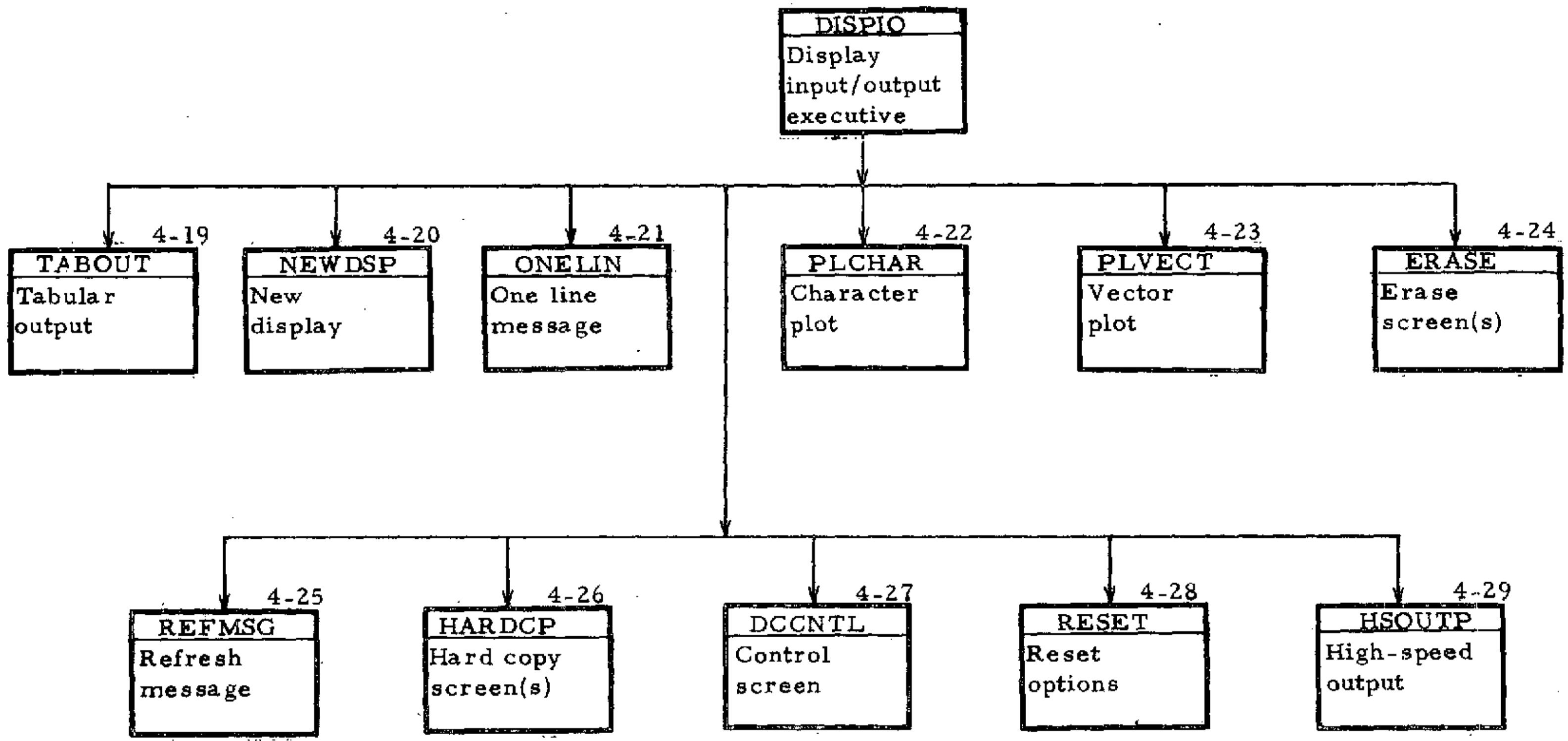


Figure 4-15

APPLICATION PROGRAM REQUEST TO DISPLAY CONTROLLER PARAMETER FORMATS

Request Code	Function	Parameters		
		1	2	3
1	Tabular Output	A(1) (1)	A(data length) (4)	A(buffer format A) (2)
2	New Display	A(2)	A(display name)	A(overlay option) (6)
3	One-Line Message	A(3)	A(data length)	A(buffer format A)
4	Character Plot	A(4)	A(data length)	A(buffer format B)
5	Vector Plot	A(5)	A(data length)	A(buffer format C)
7	Erase	A(7)	A(screen #)	
9	Refresh Message	A(9)	A(data length)	A(buffer format B)
10	Hard Copy	A(10)	A(screen #)	
11	Auxiliary Screen	A(11)	A(screen #)	
13	Reset	A(13)	A(option #) (3)	
14	High-Speed Output	A(14)	A(data length) (5)	A(buffer format A)

(1) A( ) indicates address of parenthesized element.

(2) See Figure 4-17 for formats.

(3) Parameter 2

1 = refresh message
16 = terminate to DOS
32 = temporarily return to DOS
64 = return to Display Controller

(4) All data lengths are byte lengths.

(5) Parameter zeroed when output complete; must be set before each call.

(6) 0 = new display replaces current display.
-1 = new display overlays current display.
next display = parameter 3 new display overlays parameter 2 new display which overlays current display.

Figure 4-16

Format A

Char 2	Char 1 (1)
Char 4	Char 3
Char 6	Char 5

Format B

Screen #	(2)	Plot 1
Character Size	(3)	Plot 1
Initial X	(4)	Plot 1
Initial Y	(5)	Plot 1
# Characters		Plot 1
A (Characters)	(8)	Plot 1
Screen #		Plot 2
Character Size		Plot 2
Initial X		Plot 2
Initial Y		Plot 2
# Characters		Plot 2
A (Characters)		Plot 2

Format C

Screen #	
Plot Type	(6)
X	
Y	
X	
Y	
.	(7)
.	

(1) Character = ASCII Character

(2) Screen # = 1 screen 1
2 screen 2
3 both screens

(3) Character Size = -1 for refresh vector plot message; otherwise not used.

(4) X = 0-1023

(5) Y = 0-800

(6) Plot Type = -1 solid

(7) disconnects within a plot represented by -1.

(8) a plot buffer (series of x, y points and disconnects) for refresh vector plot message.

Figure 4-17

Each parameter is set up as the address of the value being passed as the parameter. For all request types the first parameter is the address of the request code number. The second and third parameters vary according to request type.

All requests from the application program for output to the display are directed to the primary screen unless a screen number is specifiable through the parameter list. The primary screen is that screen to which the displays are being directed by the controller. The default is screen 1 (the screen with the keyboard), but is changeable through the application program Auxiliary Screen Request. Additionally, all characters output will be the standard hardware size even where specifiable through the parameter list.

Display Input/Output Executive

The Display Input/Output Executive (DISPIO (Figure 4-18)) is called directly by the application program to execute 1 of the 11 requests listed above. DISPIO determines which type of request is being made and calls the appropriate routine to process it. When the request processing is complete, DISPIO returns to the application program that called it.

Tabular Output - Request Code 1

The Tabular Output routine (TABOUT (Figure 4-19)) processes requests for the output of data to the screen to areas predefined in the Display Library fill page for the display currently on the screen.

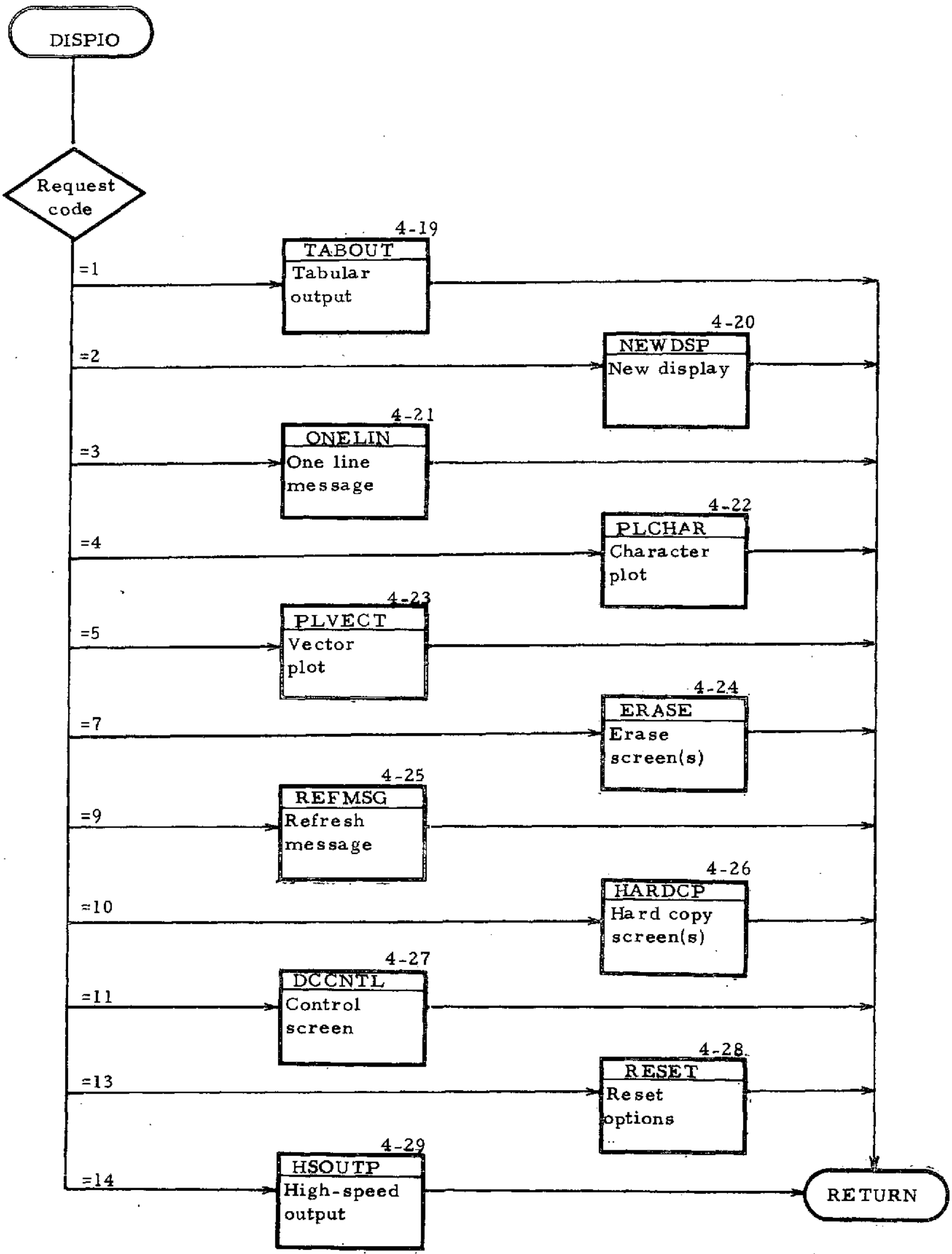
The location of the fill page for the current display within the Display Library is calculated from the Display Library index and the appropriate block is read into core from the Display Library by the LIBINP routine. Using the information in the fill page, the first fill field is set up in an output string with the data passed from the application program and sent to the display by the DSPOUT routine. This process is repeated until (1) all the data from the application program has been sent to the display or (2) all the fill fields specified in the fill page have been used. The alphanumeric cursor is then repositioned by the ALPCRS routine to the location it occupied before the tabular output was sent.

New Display - Request Code 2

The New Display routine (NEWDSP (Figure 4-20)) determines if the new display is to overlay the current display or replace it. If it is to overlay, it then determines if a third display is to overlay the second. It then brings up the appropriate displays accordingly by calling the NXTDSP routine.

One-Line Message - Request Code 3

The One-Line Message routine (ONELIN (Figure 4-21)) processes the application program request for output of a message to the left side of the bottom line of the display screen. This message is output in permanent storing mode.



DISPIO
Request code
=1
4-19
TABOUT
Tabular output
=2
4-20
NEWDSP
New display
=3
4-21
ONELIN
One line message
=4
4-22
PLCHAR
Character plot
=5
4-23
PLVECT
Vector plot
=7
4-24
ERASE
Erase screen(s)
=9
4-25
REFMSG
Refresh message
=10
4-26
HARDCP
Hard copy screen(s)
=11
4-27
DCCNTL
Control screen
=13
4-28
RESET
Reset options
=14
4-29
HSOUTP
High-speed output
RETURN

Figure 4-18

TABULAR OUTPUT

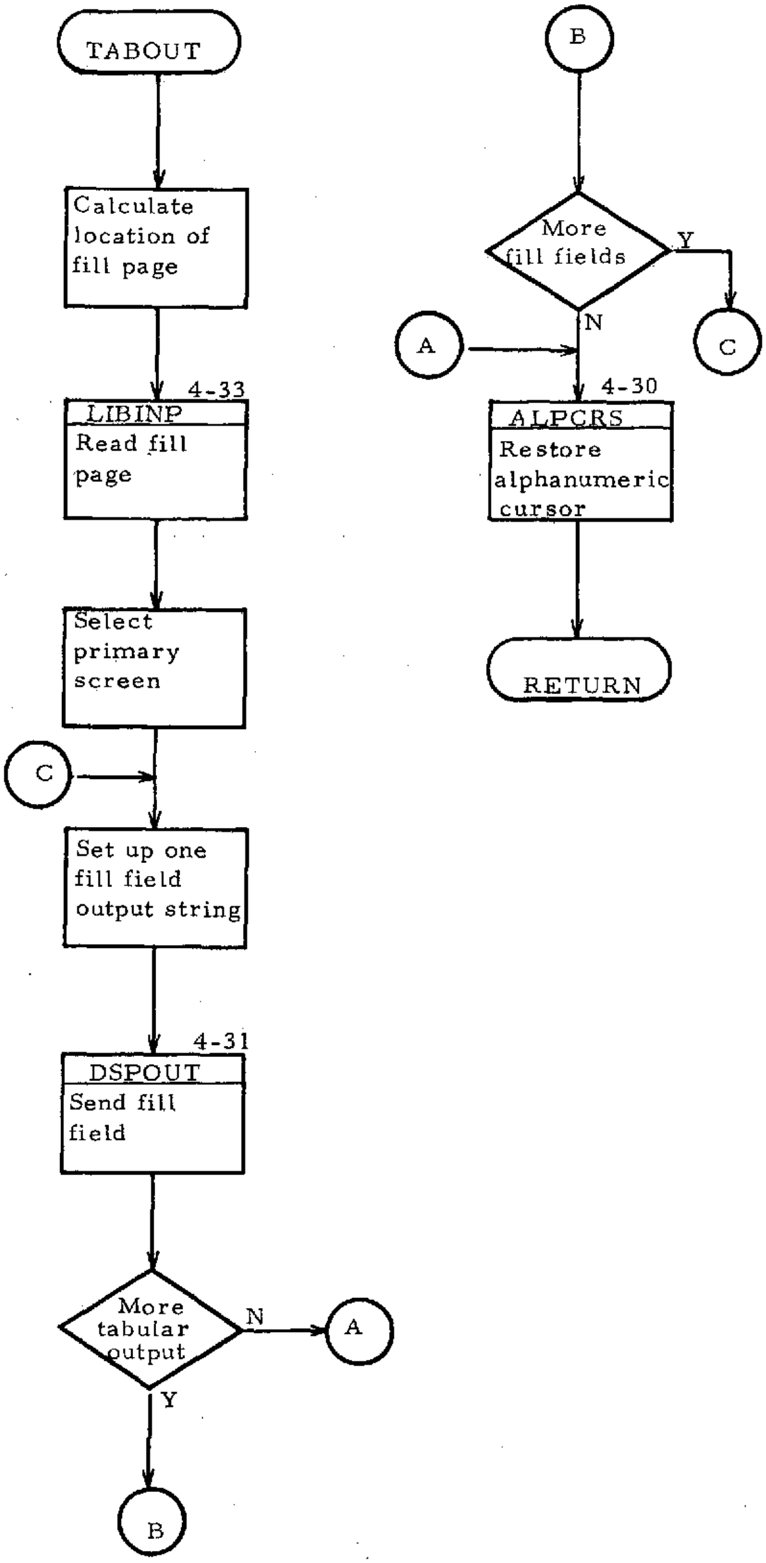


Figure 4-19

NEW DISPLAY

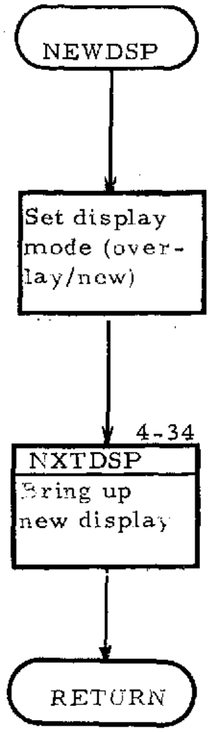


Figure 4-20

ONE LINE MESSAGE

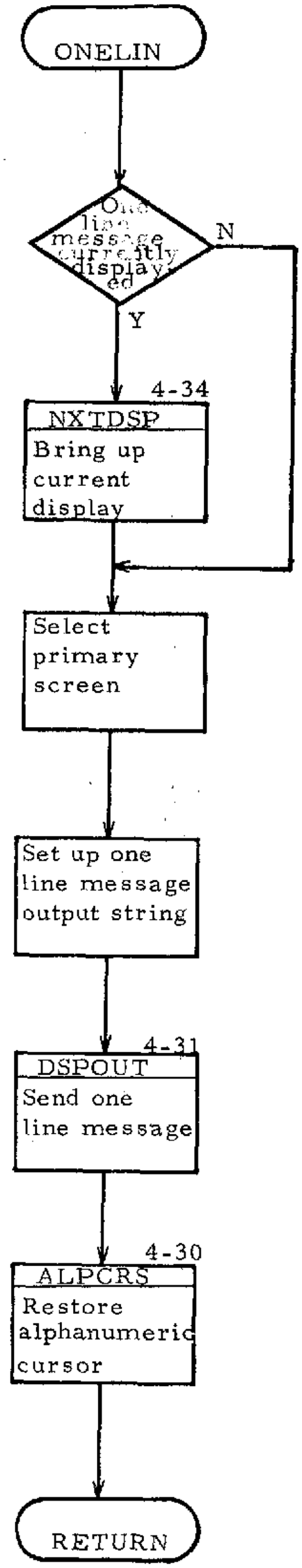


Figure 4-21

Therefore, if there already was a one-line message on the screen, a fresh copy of the current display is brought up before the newly requested message is output. The DSPOUT routine is called to perform the output. The alphanumeric cursor is then repositioned by the ALPCRS routine to the position it occupied before the one-line message was output.

Character Plot - Request Code 4

The Character Plot routine (PLCHAR (Figure 4-22)) processes the application program request for a series of alphanumeric characters to be output to any specified positions on the screen in permanent storing mode.

Each series of characters specified by the application program is set up in an output string preceded by the x-y position on the screen. The DSPOUT routine is called to output the string to the terminal and the ALPCRS routine repositions the alphanumeric cursor to its position prior to output of the character plot(s).

Vector Plot - Request Code 5

The Vector Plot routine (PLVECT (Figure 4-23)) processes the application program request for a series of vectors to be drawn to any specified positions on the screen in permanent storing mode. The output string is set up as a series of x-y points. If a disconnect is indicated by the application program, a new output string is set up to begin another series of vectors. When the entire output string is set up, the DSPOUT routine is called to send the output to the screen.

Erase Screens - Request Code 7

The Erase Screens routine (ERASE (Figure 4-24)) processes the application program request to erase either or both screens. The erase command string is set up for the requested screen(s) and DSPOUT routine is called to output the command string.

Refresh Message - Request Code 9

The Refresh Message routine (REFMSG (Figure 4-25)) processes the application program request for a series of alphanumeric messages or vector plots to be output to any specified positions on the screen in non-storing or write-through mode.

The output string is set up as a series of character plots and vector plots except they are specified in write-through mode. This routine is only responsible for setting up the output string. The actual output is done by the Refresh Screen routine (LOOP20 (Figure 4-14)).

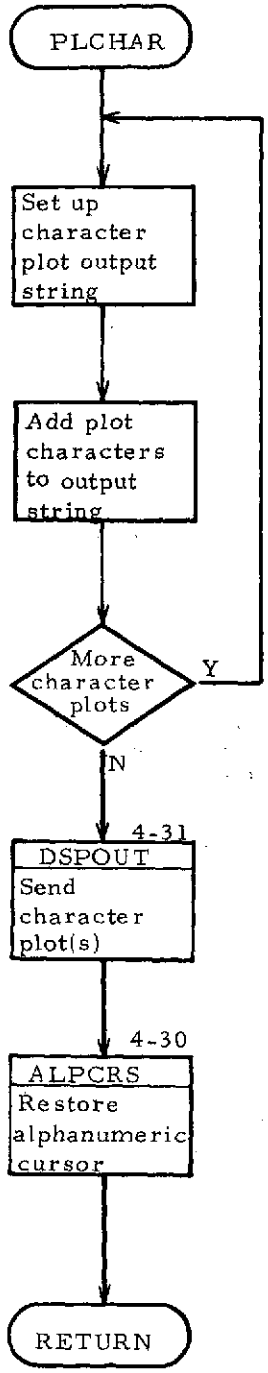

PLCHAR
Set up character plot output string
Add plot characters to output string
More character plots
Y
N
4-31
DSPOUT
Send character plot(s)
4-30
ALPCRS
Restore alphanumeric cursor
RETURN

Figure 4-22

VECTOR PLOT

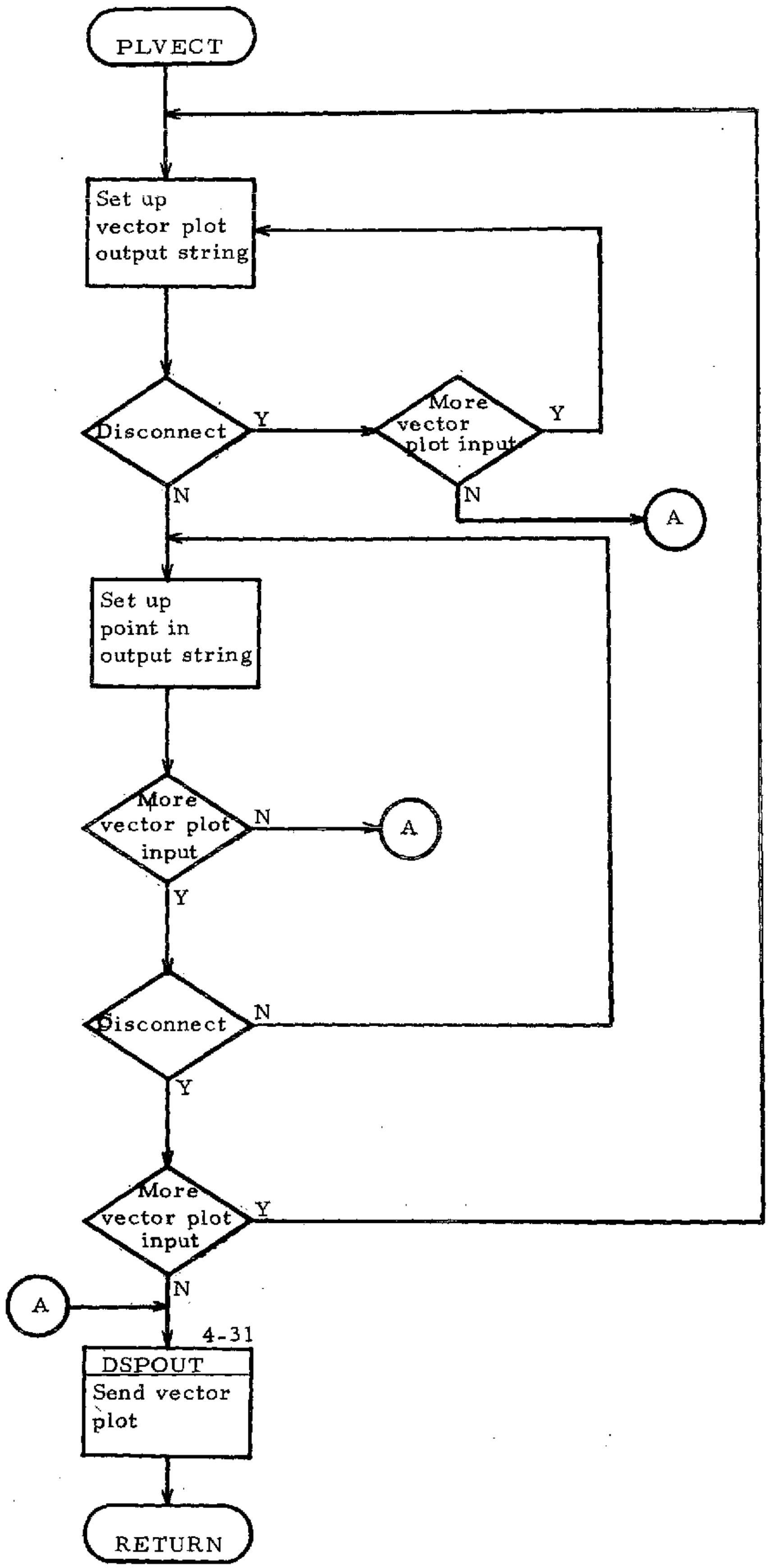


Figure 4-23

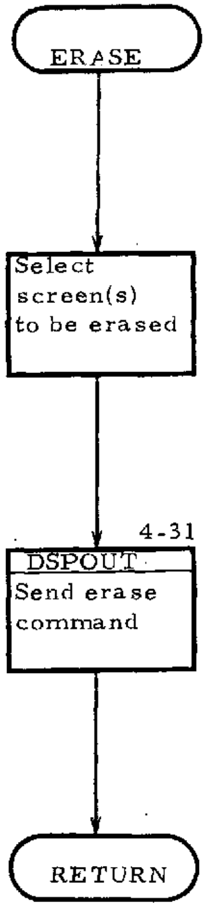

ERASE
Select screen(s) to be erased
4-31
DSPOUT
Send erase command
RETURN

Figure 4-24

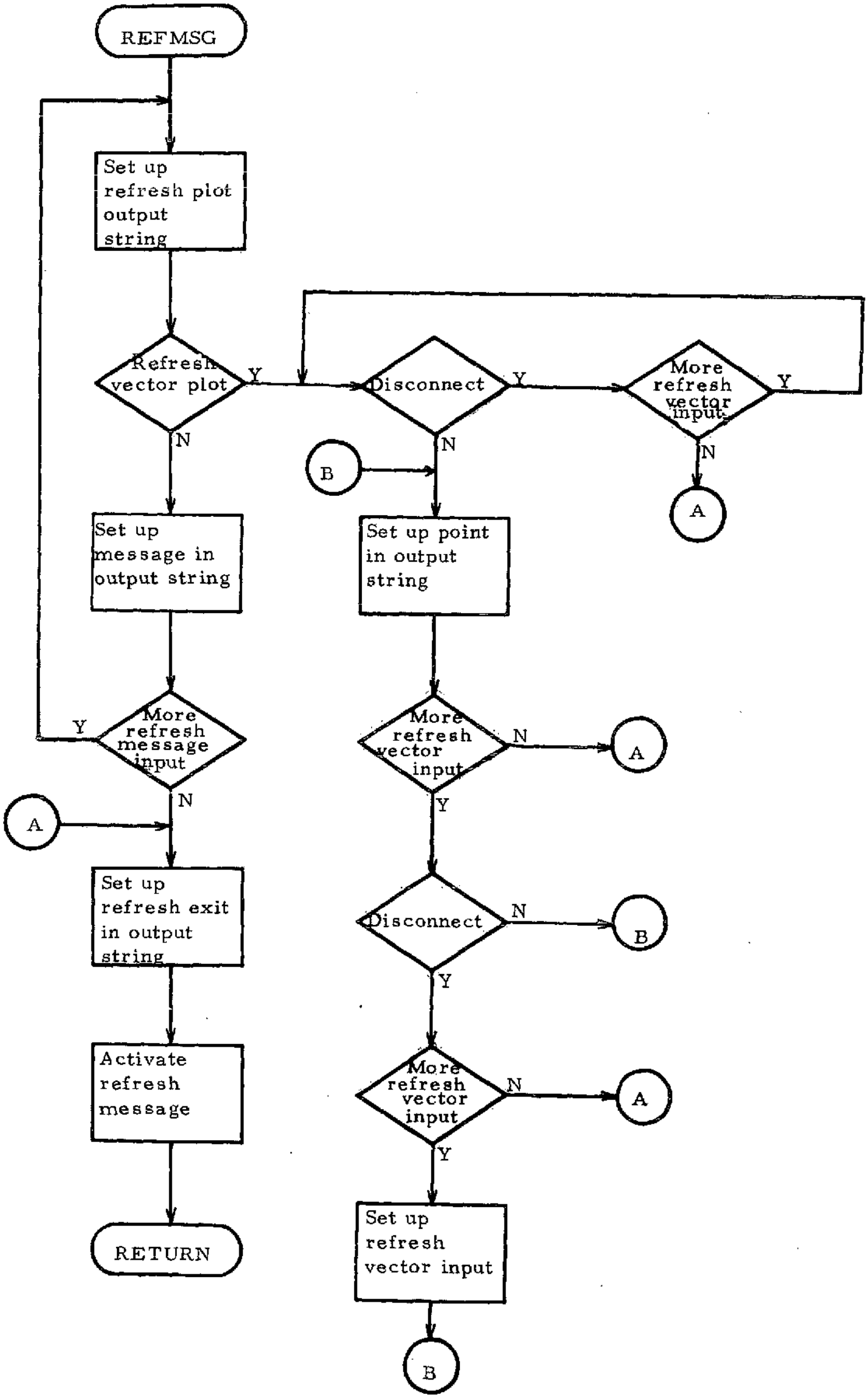

REFMSG
Set up refresh plot output string
Refresh vector plot
Y
N
Disconnect
Y
N
More refresh vector input
Y
N
A
B
Set up message in output string
Set up point in output string
More refresh message input
Y
N
More refresh vector input
N
Y
A
A
Set up refresh exit in output string
Disconnect
N
Y
B
Activate refresh message
More refresh vector input
N
Y
A
RETURN
Set up refresh vector input
B

Figure 4-25

Hard Copy Screens - Request Code 10

The Hard Copy Screens routine (HARDCP (Figure 4-26)) processes the application program request to hard copy either or both screens. The hard copy command string is set up for the requested screen(s) and the DSPOUT routine is called to output the command string.

Auxiliary Screen - Request Code 11

The Auxiliary Screen routine (DCCNTL (Figure 4-27)) processes the application program request to establish a specific screen to be the primary screen (the screen to which all display information is directed).

Reset Options - Request Code 13

The Reset Options routine (RESET (Figure 4-28)) processes the application program request to reset certain conditions established for Display Controller processing. These options are:

- o Deactivate refresh messages
- o Terminate execution - return to DOS
- o Temporarily return to DOS
- o Return to Display Controller

High-Speed Output - Request Code 14

The High-Speed Output routine (HSOUTP (Figure 4-29)) processes the application program request for output of preformatted data directly to the display screen.

If a previous output request is still being processed, the currently requested output is stacked in a Display Controller table for later output. If there is no output in progress, the output of this requested data is set up and begun. The first byte of data is moved to the output register and the output interrupt is enabled. The remainder of the data from the request is output on an interrupt basis by the OUTINT routine.

4.3 Common Routines

There are several routines that perform general functions required by both the User Input Processor (Section 4.1) and the Application Program Request Processor (Section 4.2). These routines are:

- o Position Alpha Cursor - ALPCRS

HARD COPY SCREENS

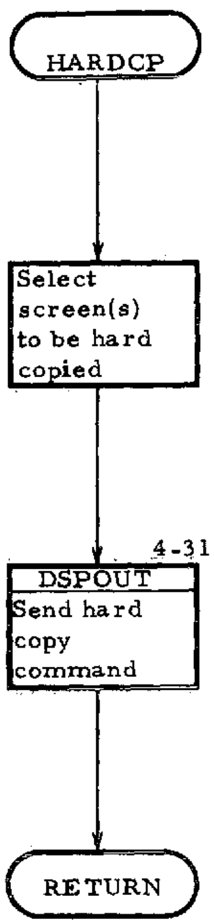


Figure 4-26

AUXILIARY SCREENS

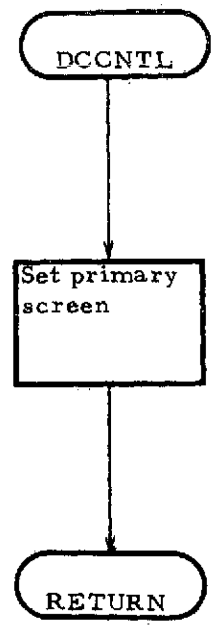


Figure 4-27

RESET OPTIONS

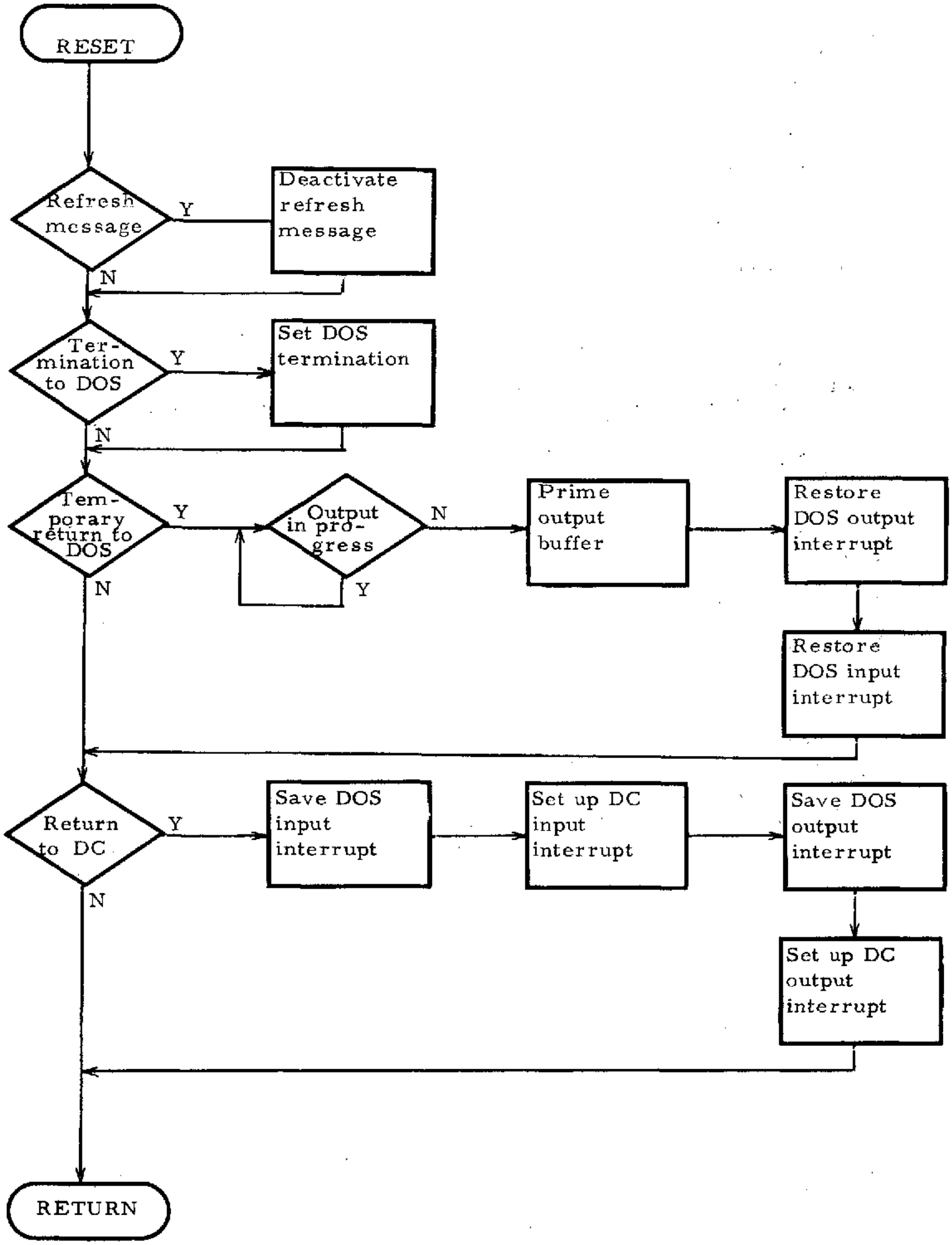


Figure 4-28

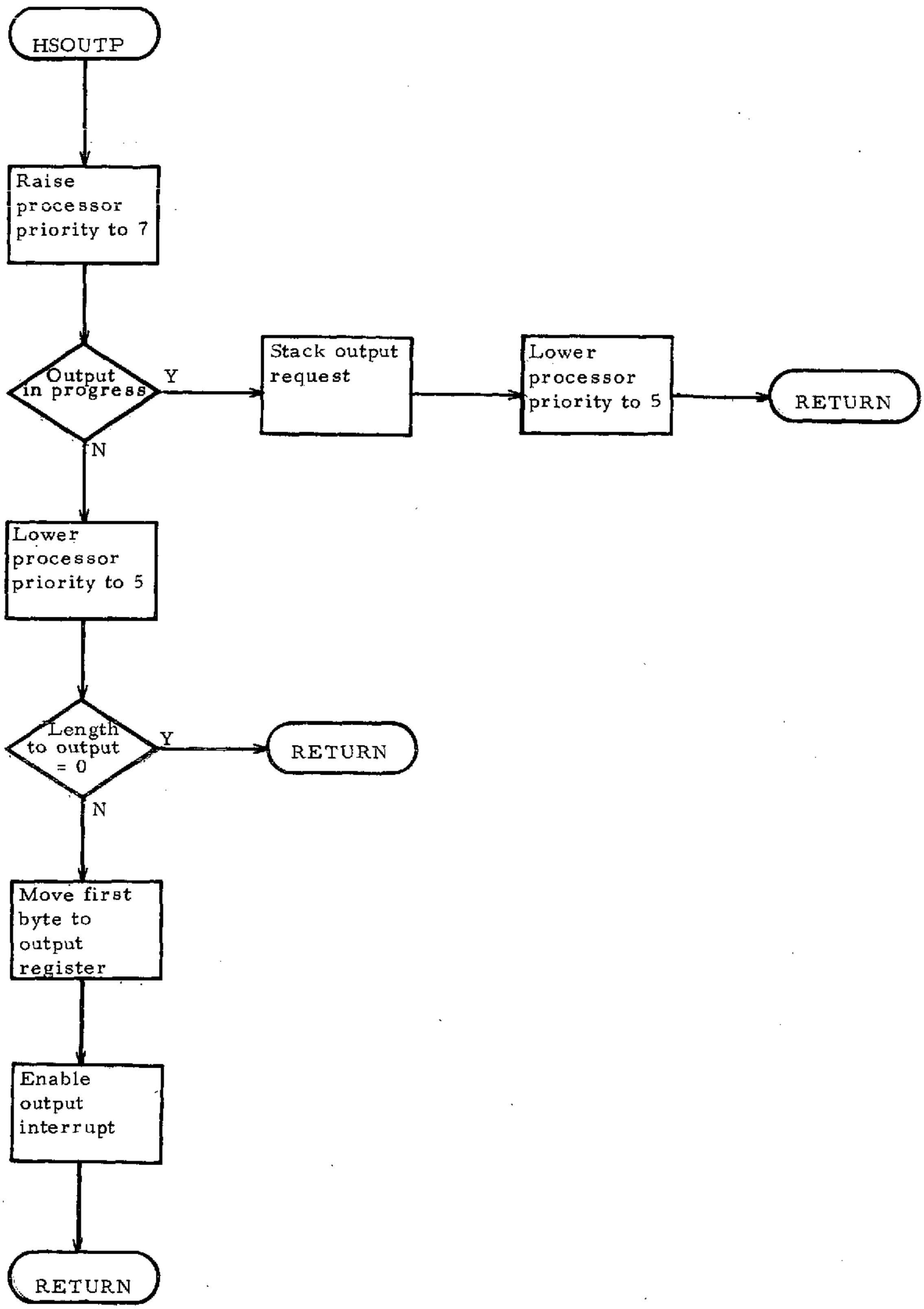

HSOUTP
Raise processor priority to 7
Output in progress
Y
Stack output request
Lower processor priority to 5
RETURN
N
Lower processor priority to 5
Length to output = 0
Y
RETURN
N
Move first byte to output register
Enable output interrupt
RETURN

Figure 4-29

- o Send Command String to Display - DSPOUT
- o Call Next Program - INVPGM
- o Read Display Library - LIBINP
- o Bring Up Next Display - NXTDSP
- o Set Up First Key-In Field - KEYSET

Position Alpha Cursor

The Position Alpha Cursor routine (ALPCRS (Figure 4-30)) sends the alphanumeric cursor to the x-y position set up in the output string by the calling program.

Send Command String to Display

The Send Command String to Display routine (DSPOUT (Figure 4-31)) processes all output requests whose output strings are formatted by the Display Controller (i.e., all output except High-Speed Output requests from the application program). This routine sets the output up as a High-Speed Output request and calls DISPIO to process it as such.

Call Next Program

The Call Next Program routine (INVPGM (Figure 4-32)) calls the application program to execute the next program indicated in the parameter list (Figure 4-12) set up by the routine that called INVPGM. INVPGM initiates the data tablet to send-point mode so that the application program processing will not be interrupted by data tablet input. INVPGM calls the Executive (ROOTEX) which in turn executes the next program.

When the next program has completed processing, INVPGM reestablishes burst mode for data tablet input.

Read Display Library

The Read Display Library routine (LIBINP (Figure 4-33)) reads the Display Library block number as set up by the routine that called LIBINP.

Bring Up Next Display

The Bring Up Next Display routine (NXTDSP (Figure 4-34)) processes the request for a display to be presented on the screen. DSPOUT is called first to erase the primary screen unless the next display is to overlay the current display. If the display to be brought up is specified as PREV, the display history table is accessed to determine the name of the previous display. If the next display

POSITION ALPHA-NUMERIC CURSOR

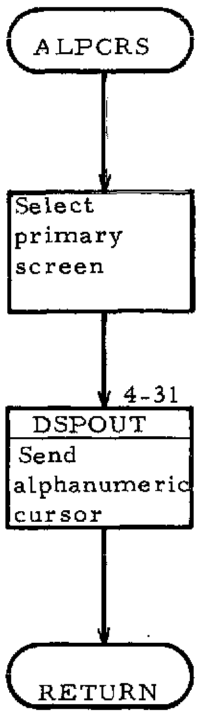


Figure 4-30

SEND COMMAND STRING TO DISPLAY

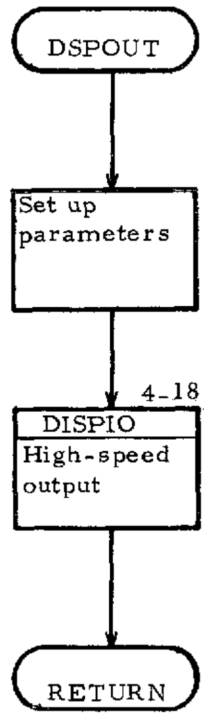


Figure 4-31

CALL NEXT PROGRAM

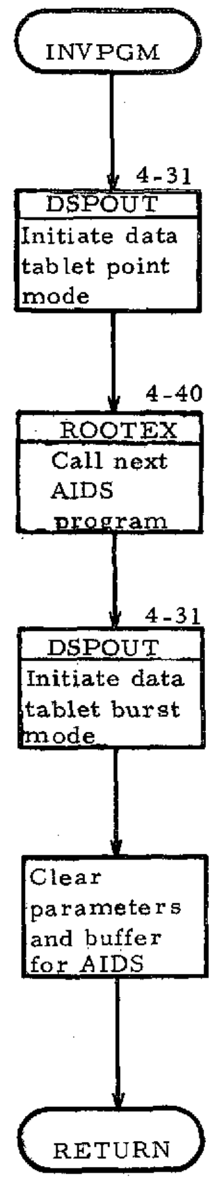


Figure 4-32

READ DISPLAY
LIBRARY

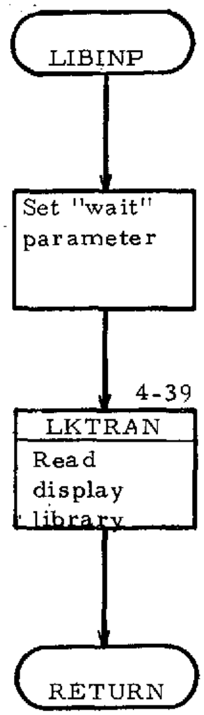


Figure 4-33

BRING UP NEXT DISPLAY

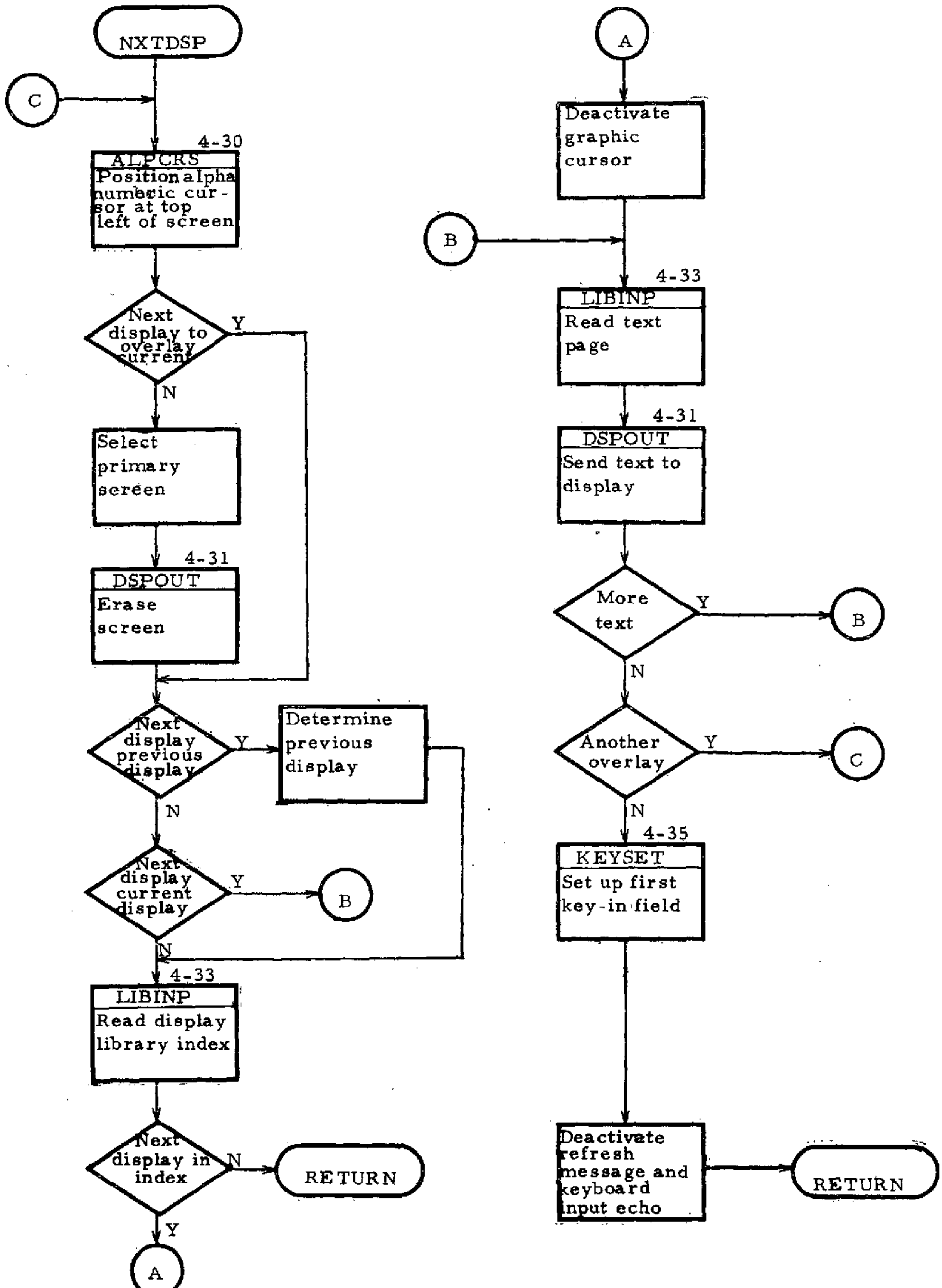


Figure 4-34

is not the current display, the Display Library index is read by LIBINP and then searched for the next display entry. (If the next display is the current display, its index entry is already in core.)

Using the index, the location of the text page is determined. The text page(s) is read by LIBINP and output to the screen by DSPOUT. If a second overlay has been requested, this entire sequence is repeated.

KEYSET is called to set up for the first key-in field, and position the alphanumeric cursor accordingly. The graphic cursor, the refresh message, and the keyboard input echo are all deactivated.

Set Up First Key-In Field

The Set Up First Key-In Field routine (KEYSET (Figure 4-35)) processes the setup of the first key-in field in the current display (if one is present). The routine first reads in the key page for the current display from the Display Library through LIBINP. Using the key page information, this routine sets up appropriate counters and pointers and then positions the alphanumeric cursor at the beginning of the first key-in field.

4.4 Input/Output Interrupt Processors

The Input/Output Interrupt Processors perform all I/O to the terminal through the input/output hardware registers. They both process on an interrupt driven basis.

Output Interrupt Processor

The Output Interrupt Processor (OUTINT (Figure 4-36)) moves the next/first byte of output into the output register. (The High-Speed Output routine(HSOUTP (Figure 4-29)) is responsible for initiating the output process and stacking waiting requests.)

When the OUTINT routine determines that all bytes have been sent for the current request, it sets the output string length in the output requestor's area to zero to indicate completion. If another output request is waiting, it removes it from the wait stack and moves the first byte of the output string into the output register, thus initiating the next request.

All entries to this routine are through the output interrupt address set up by the Initialization routine (INIT (Figure 4-4)).

Input Interrupt Processor

The Input Interrupt Processor (INPINT (Figure 4-37)) processes all input received through the input register and sets it up for further processing by the User Input Processor (Section 4.1). This input includes data tablet input and keyboard input both received one byte at a time.

SET UP FIRST KEY-IN FIELD

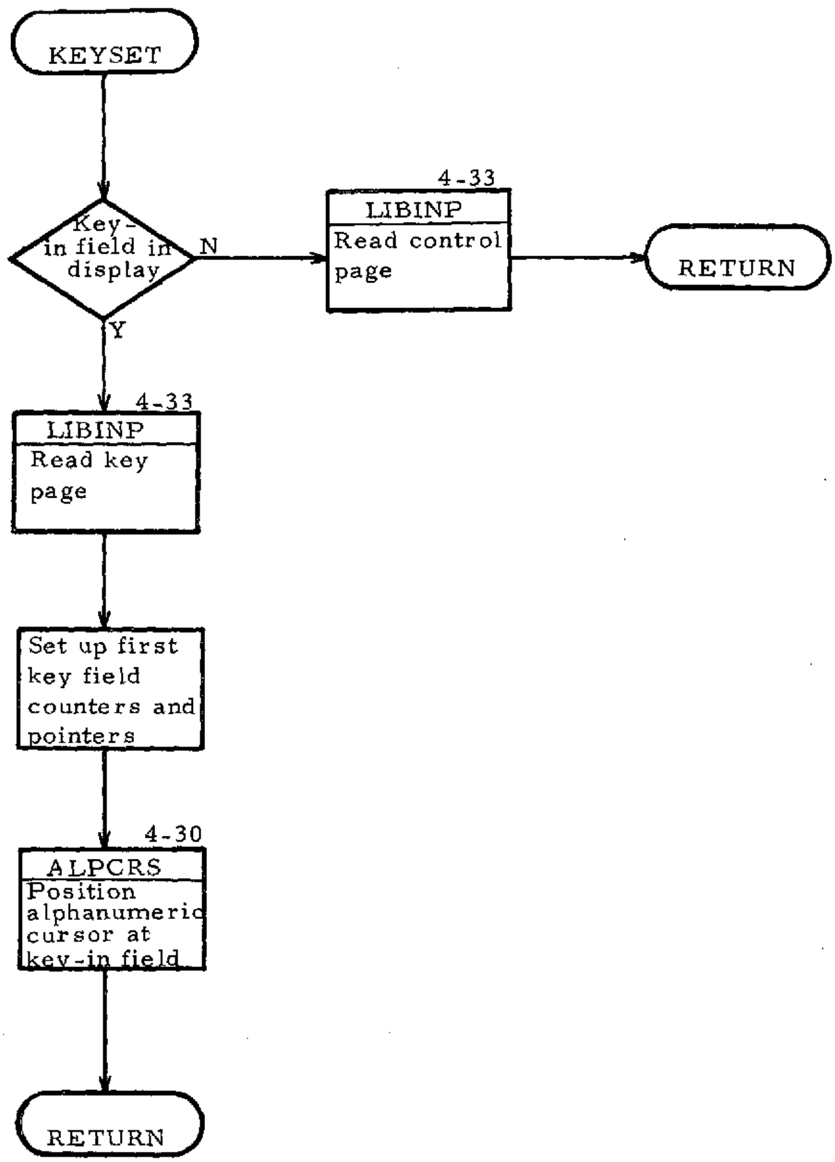


Figure 4-35

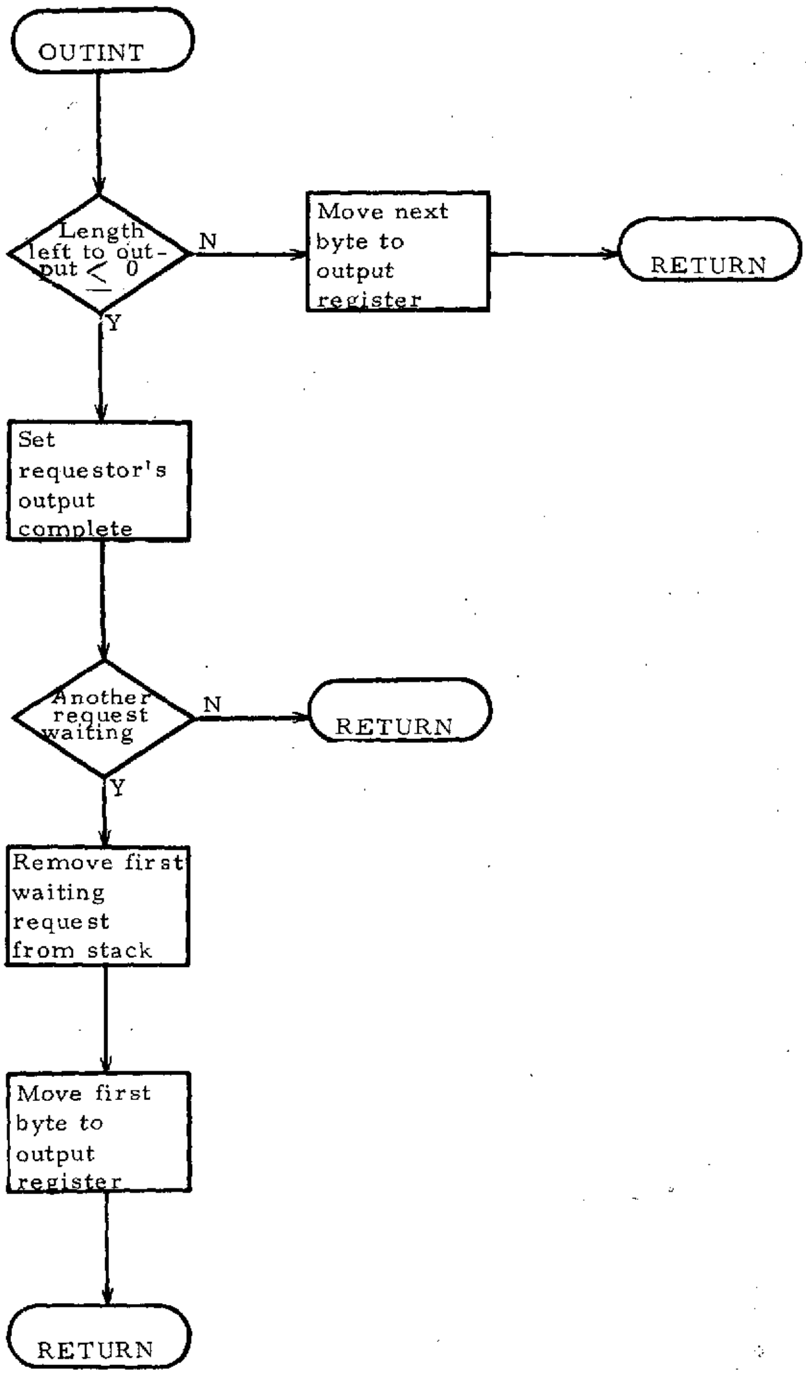

OUTINT
Length left to output ≤ 0
N
Y
Move next byte to output register
RETURN
Set requestor's output complete
Another request waiting
N
Y
RETURN
Remove first waiting request from stack
Move first byte to output register
RETURN

Figure 4-36

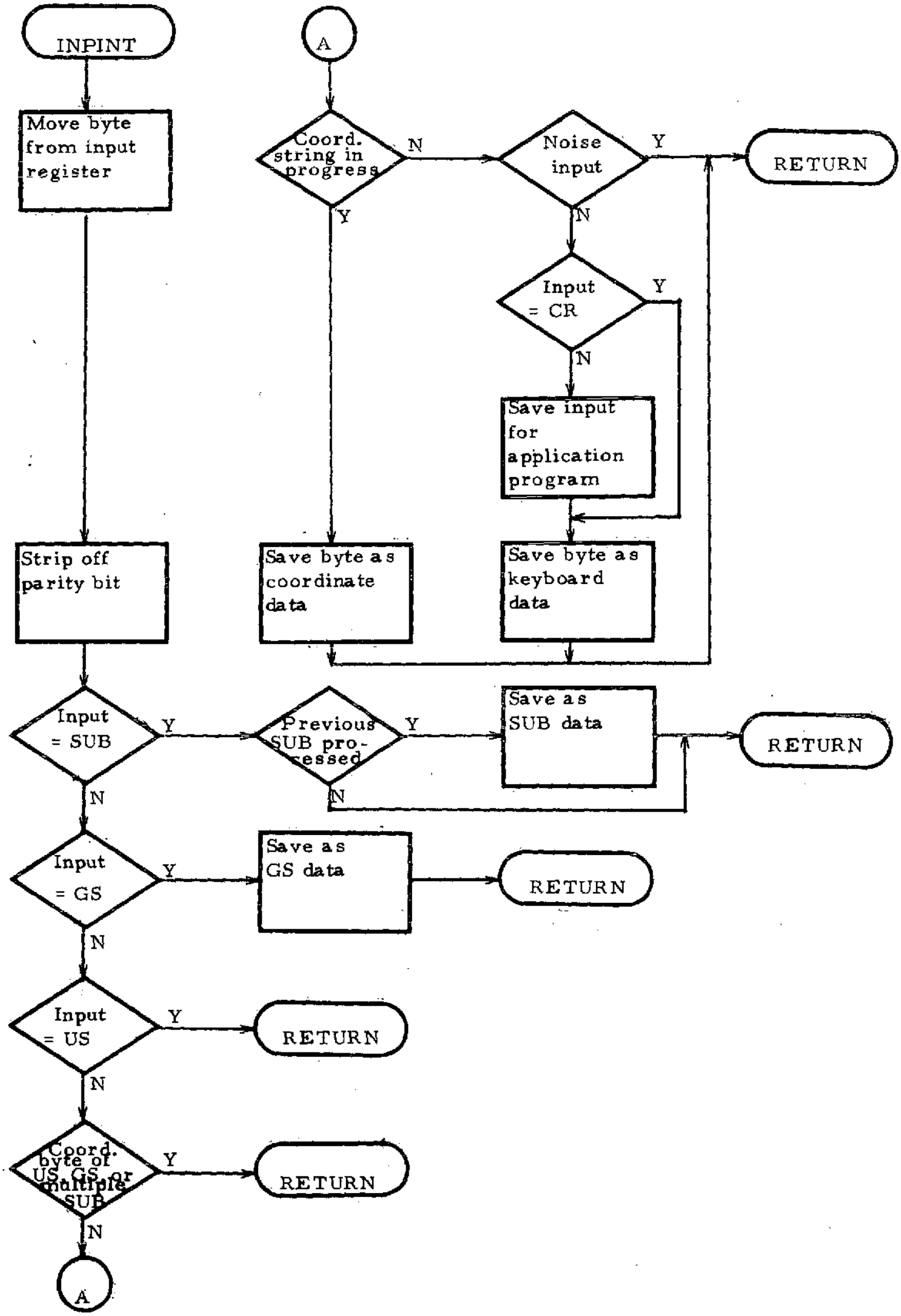

INPINT
Move byte from input register
Strip off parity bit
Input = SUB
Y
N
Input = GS
Y
N
Input = US
Y
N
Coord. byte of US, GS, or multiple SUB
Y
N
A
A
Coord. string in progress
N
Y
Noise input
Y
N
Input = CR
Y
N
Save input for application program
Save byte as keyboard data
Save byte as coordinate data
RETURN
Previous SUB processed
Y
N
Save as SUB data
RETURN
Save as GS data
RETURN
RETURN
RETURN

Figure 4-37

Data tablet input is received in a series of five bytes, a command byte followed by four coordinate bytes. The command byte is either a SUB, GS, or US. From data tablet input only the SUB command, its coordinates, and the GS command are saved for further processing. All other bytes are ignored.

If the input is not part of one of the valid five byte input strings mentioned above and determined not to be noise input, it is saved as keyboard input.

All entries to this routine are through the input interrupt address set up by the Initialization routine (INIT (Figure 4-4)).

4.5 Disk I/O Handling

The two routines in this area are called to perform all I/O to the disk.

Initialize Data Set Device

The routine (INITDS (Figure 4-38)) which performs this function issues a DOS system macro to initialize the specified device to insure that the device driver is in core for subsequent I/O operations.

Read/Write Data Set

This routine (LKTRAN (Figure 4-39)) performs the basic function of reading and writing records from and to data sets on disk. It performs error checking prior to and following the I/O function performed. This checking is done to determine if the data set to be read or written exists, if the record number to be read is valid, and if any error occurred on the I/O operation. A return parameter is set accordingly.

4.6 Executive

The Executive (ROOTEX), as shown in Figure 4-40, directs the input from the display to the appropriate application "next" program. After determining that the indicated next program is a valid program, ROOTEX gives control to the required routine.

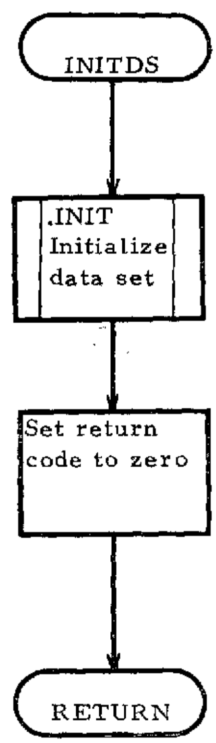

INITDS
.INIT
Initialize
data set
Set return
code to zero
RETURN

Figure 4-38

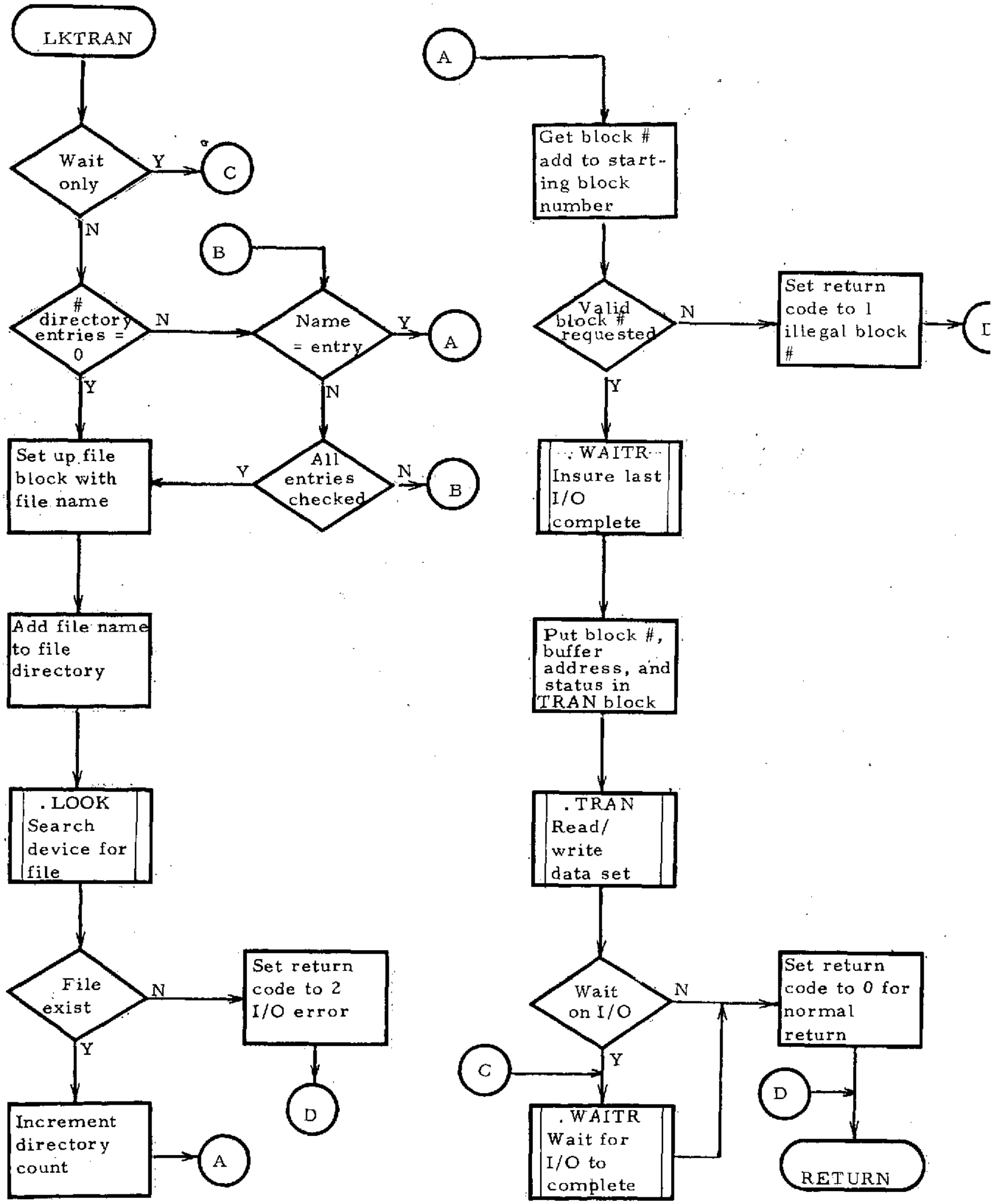

LKTRAN
Wait only
Y
C
N
B
directory entries = 0
N
Name = entry
Y
A
Y
N
Set up file block with file name
Y
All entries checked
N
B
Add file name to file directory
. LOOK Search device for file
File exist
N
Set return code to 2 I/O error
Y
D
Increment directory count
A
A
Get block # add to starting block number
Valid block # requested
N
Set return code to 1 illegal block #
D
Y
. WAITR Insure last I/O complete
Put block #, buffer address, and status in TRAN block
. TRAN Read/ write data set
Wait on I/O
N
Set return code to 0 for normal return
C
Y
D
. WAITR Wait for I/O to complete
RETURN

Figure 4-39

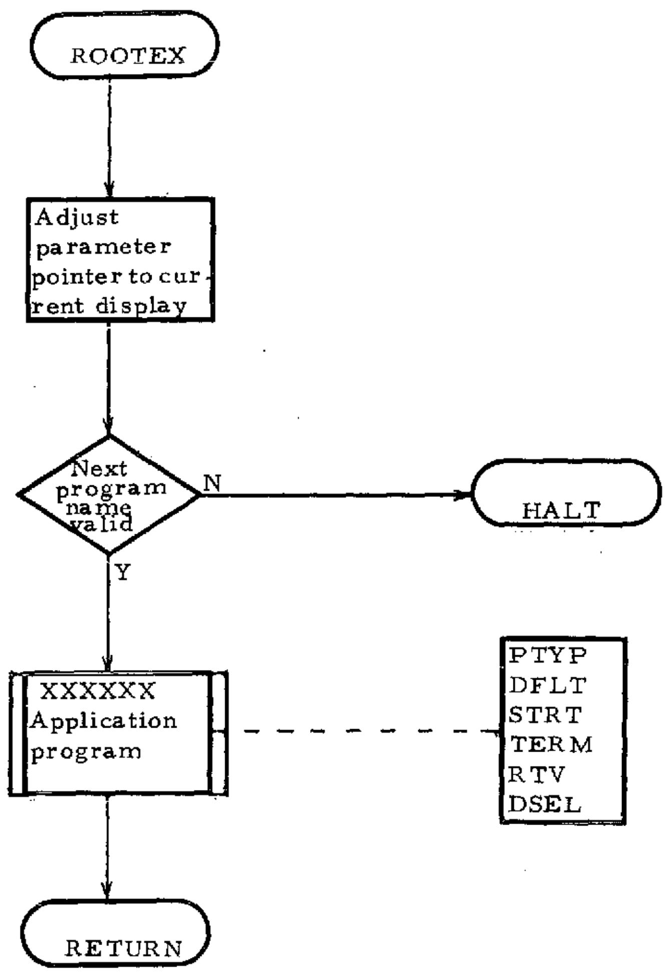

ROOTEX
Adjust parameter pointer to current display
Next program name valid
N
HALT
Y
XXXXXX Application program
PTYP
DFLT
STRT
TERM
RTV
DSEL
RETURN

Figure 4-40

5. DISPLAY LIBRARIAN

To minimize the on-line core and time requirements necessary to create each individual application program oriented display and to provide a completely general graphics capability, all displays are preformatted by an off-line Display Librarian. The librarian accepts card images of the text and control information defining each display and creates a "book" of displays.

The display book resides on disk and contains a display chapter for each display within the book. A display index is generated by the librarian defining the location of each display chapter within the display book. Each chapter is further sub-divided into two "pages":

- Text Page
- Control Page

The text page of each display chapter contains display text information in an expanded format consisting of embedded graphic control commands. The text page exists in a format that is ready for immediate generation on the display screen and requires no editing, scanning, or unpacking in real time. The control page is made up of the pen, keyboard, and fill pages that provide the control information needed by the real time Display Controller to respond to tablet pen and keyboard inputs and application program fill-in requests.

During real time operation the Display Controller, upon detecting a tablet pen selection or a keyboard input, uses the control information associated with the display being viewed to determine the user specified action to be taken.

The primary purpose of the Display Librarian is the creation of the preformatted display book from user defined input. To insure that the data can be correctly displayed and operated on during real time operations, it is necessary for the Display Librarian to perform extensive error checking on the user's input data prior to creating the display chapter on disk. The librarian can therefore serve as a display assembler and aid the user in defining his displays. During the processing of a display, records that contain errors are listed along with messages describing the errors. Each display must be completely free of errors before it is added to the display book on disk.

5.1 Display Book

The display book is a sequentially organized contiguous file on disk consisting of the display index and a display chapter for each display within the book. Figure 5-1 presents the process by which the display book is generated and the organization of the display chapters and the display index on disk.

5.1.1 Display Index

The display index is the first record of the display book and defines the location of each display chapter within the display book file. The display index is segmented into 256-word blocks; the format of the index is presented in Figure 5-2. If more than one index record is necessary to define the display chapters, i.e., there are more than 50 displays in the "book," additional index records are placed at the end of the display chapters. The first word of each index record contains the relative block number of the next index record.

The relative block number of the display chapter is the relative block number within the display book file of the first text page block. The blocks within a display chapter are organized sequentially as text blocks followed by control blocks. If any of the blocks are not required for a display, the appropriate display index entry is set to zero. The display name is an integer between 0001 and 9999 defined by the user's input card.

The Display Controller reads the display index blocks, locates the appropriate display chapter index by virtue of the display name, and then uses the relative block number of the display chapter to access the display within the display book file.

5.1.2 Display Chapters

The display chapters are divided into two "pages": the text page(s) and the control page(s). The contrbl page(s) is made up of the pen, keyboard, and fill pages. The text and control pages are segmented into 256-word blocks.

Text Page

The text page contains the information that is to be displayed to the operator. This information consists of embedded graphic orders, character control orders, alphanumeric information and special symbols that have meaning to the operator and the Display Controller. The "#" symbol defined by the user input indicates locations where an application program may fill-in

DISPLAY GENERATION

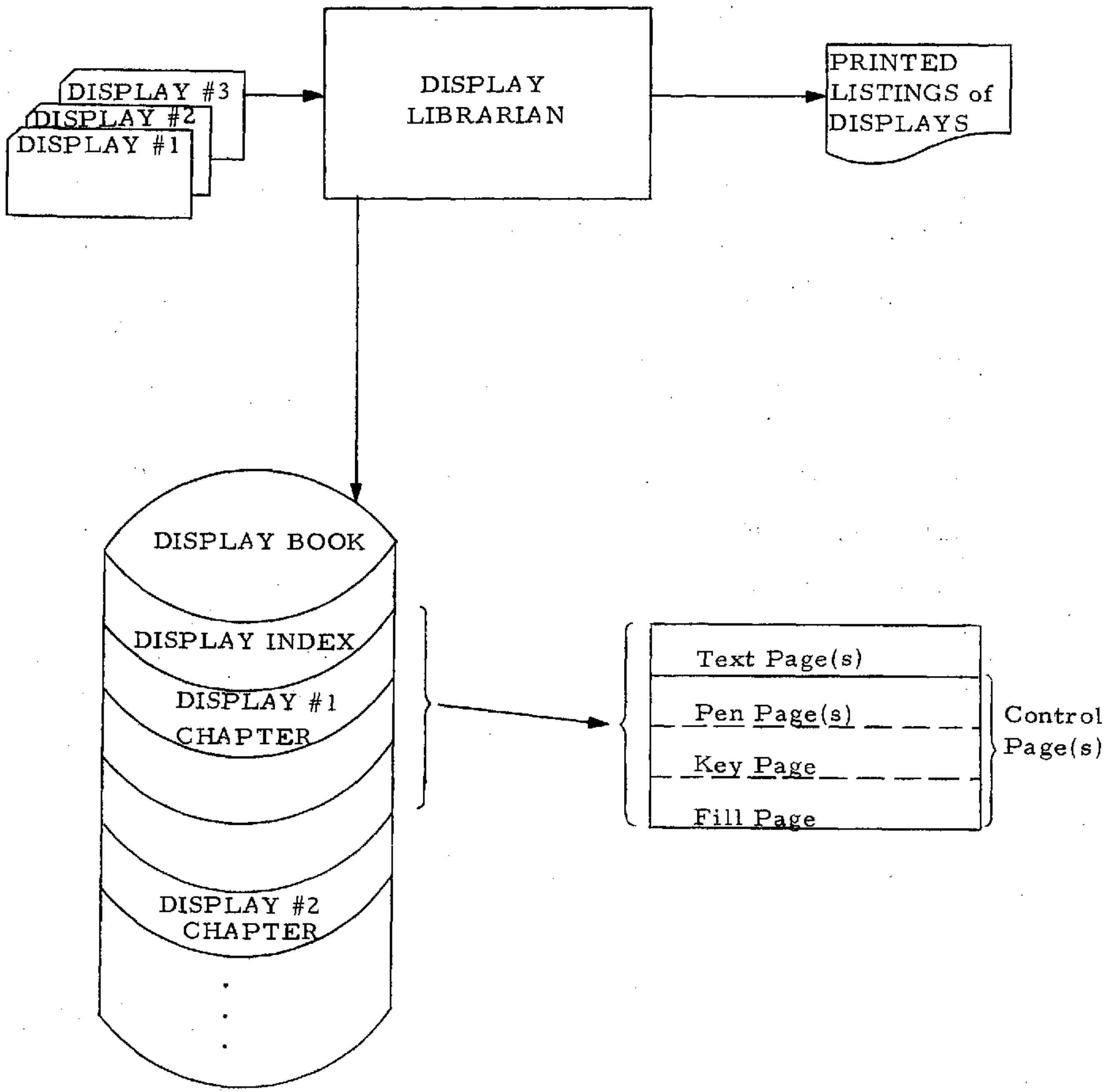


Figure 5-1

DISPLAY INDEX FORMAT

16-bit word

Relative Block # of Next Index Record	
# of Displays in This Block	
Display Name	
Relative Block # of First Text Page	
Relative Block # of First Control Page	Index Entry
Pointer to Keyboard Page	
Pointer to Fill Page	
⋮	

Figure 5-2

tabular data. The "#" symbol is replaced by a blank within the text page when a display is presented to eliminate the need to refresh the entire display picture when an application sends data to the screen. The data is displayed in the format initially defined by the "#" symbols. The "|" symbol indicates keyboard input areas and is replaced by an underline (_) character within the text page and display picture. The pen option areas are defined to the Librarian as the area between the two enclosing symbols "<" and ">." The characters within the symbols are displayed on the display picture and comprise one pen option area.

In addition, four character sizes are supported by the Display Librarian and character control orders are defined within the text page to display the different sizes. Any combination of the character sizes may be defined for a display and the Librarian will insure the correct spacing both horizontally and vertically. Although multiple character sizes are supported by the Librarian, they are only functional when the hardware capability exists.

Control Pages

The pen, keyboard, and fill pages are grouped together under the general category of control pages because they supply the control information used by the Display Controller during real time operation to process inputs to the respective fields.

Pen Page

The pen page of the display chapter contains the control information necessary to define the areas of the display text that may be selected with the graphics tablet pen. The format of the pen page is depicted in Figure 5-3. Each pen entry is 7-words in length consisting of:

- the X-, Y-coordinate of the first character within the pen field,
- the Δ X and Δ Y of the last character within the pen field,
- the next display to be presented to the user when this pen field option is selected with the tablet pen, and
- an optional 4-character application program name to be given control when this pen field is selected.

The Display Controller determines the dimensions of the pen field from the starting X-, Y-coordinates of the first character and the Δ X and

PEN PAGE FORMAT

16-bit word

# Pen Fields in the Display	
X-Coordinate for Start of Field	
Y-Coordinate for Start of Field	
Δ X to End of Field	PEN ENTRY
Δ Y to End of Field	
Next Display Name	
4-Character Next Program Name	
. . . . . .	

Figure 5-3

Δ Y of the last character in the field. When the tablet pen input is received by the Display Controller, its coordinates are checked against the dimensions of each pen field within the pen page to determine which option was selected. When the selected option is found, the next display and the next program associated with the pen field are displayed and executed.

Keyboard Page

The keyboard page contains the following control information for each compose field within the display text (see Figure 5-4):

- o the total number and size of characters within the compose field,
- o the X-, Y-coordinate of the first character within the field,
- o the next display name to be presented to the operator, and
- o the next program name to receive the compose data.

The Tektronix keyboard permits the user to enter alphanumeric characters into computer storage for transmission to the application program. The cursor keys on the keyboard control the compose field where the data will be placed. As each character is entered, it is displayed to the operator in one of the character slots indicated by the underline (_) character for verification and editing. After entering the data, the user presses the transmission key to pass the data to the application program associated with the compose field.

Fill Page

All areas of the display text that are available for application program tabular data output must be predefined to the Display Librarian by the special symbol "#." The librarian constructs a fill page entry for each of these areas defining their location within the display text. Each fill entry is delimited by either a non-# symbol or a new display line. The format of the fill page is depicted in Figure 5-5.

5.2 Librarian Processing Flow

In creating the user's display book, the Display Librarian program executes five levels of processing. These five levels include:

1. Control card processing

KEYBOARD PAGE FORMAT

16-bit words

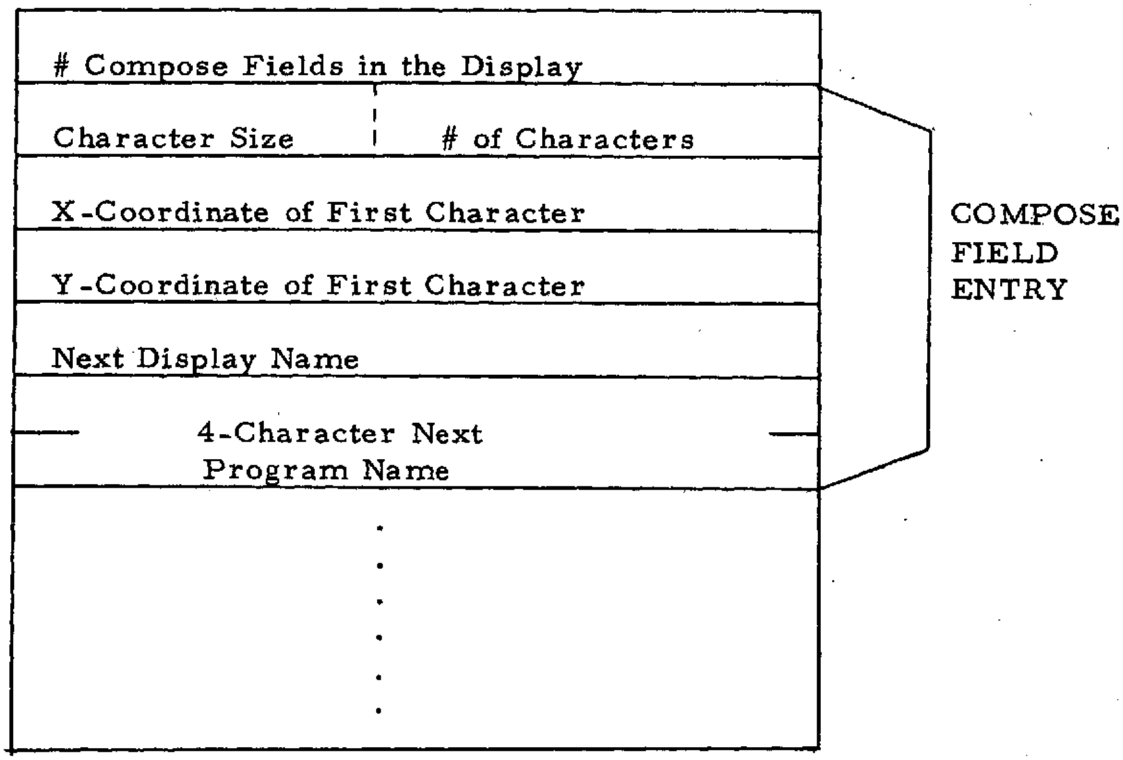


Figure 5-4

FILL PAGE FORMAT

16-bit words

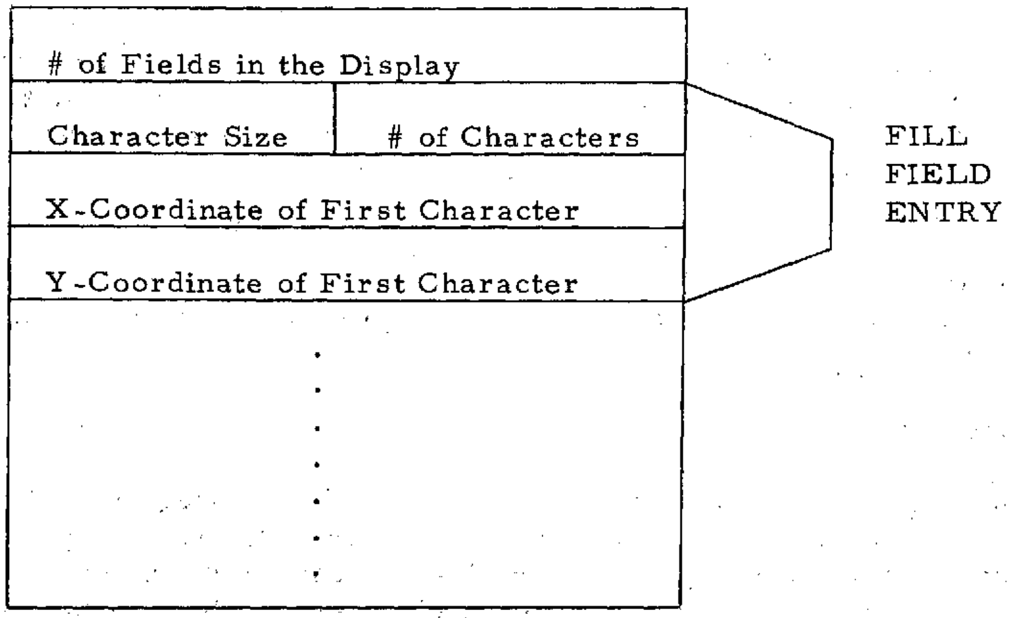


Figure 5-5

2. Pen, compose, and line card processing

3. Text card processing

4. Output formatting

5. Cleanup

Levels two, three, and four are executed for each display being defined whereas level one is executed only when display control cards are detected in the input stream and level five only upon receiving an end-of-run indication. Figures 5-6 and 5-7 present a general flow of the Display Librarian from job initiation to job end.

The following subsections describe briefly and depict in flowcharts the processing done by the librarian program on each of the above mentioned levels.

5.2.1 Control Card Processing

The output option card and delete cards are processed on this level (Figure 5-8). If an output option card is received, the appropriate option flags are set; if the update option ("U") is specified, the index of the old display book is read and maintained in core. The program then returns to read the next input card.

If the control card read is a delete, the display name specified in the card is compared against the display names in the index until the display is found or all entries are checked. If the display is found, the index entry for the display is deleted and the message DISPLAY xxxx HAS BEEN DELETED FROM THE LIBRARY is written to the printer. If the display name is not in the index, the message DISPLAY xxxx NOT FOUND IN LIBRARY is written.

5.2.2 Pen, Compose, and Line Card Processing

It is on this level of processing (Figure 5-9) that the librarian program begins building the pen and key entries (Figures 5-3 and 5-4) in the display control page(s).

In processing a pen card, the names of the next display and next program are stored in the pen page. If the X-, Y-coordinate positions are present on the input record, the ΔX and ΔY are calculated and all are placed in the pen page completing the entry for the pen field. In the cases where the X-, Y-coordinates are not on the pen card, the pen

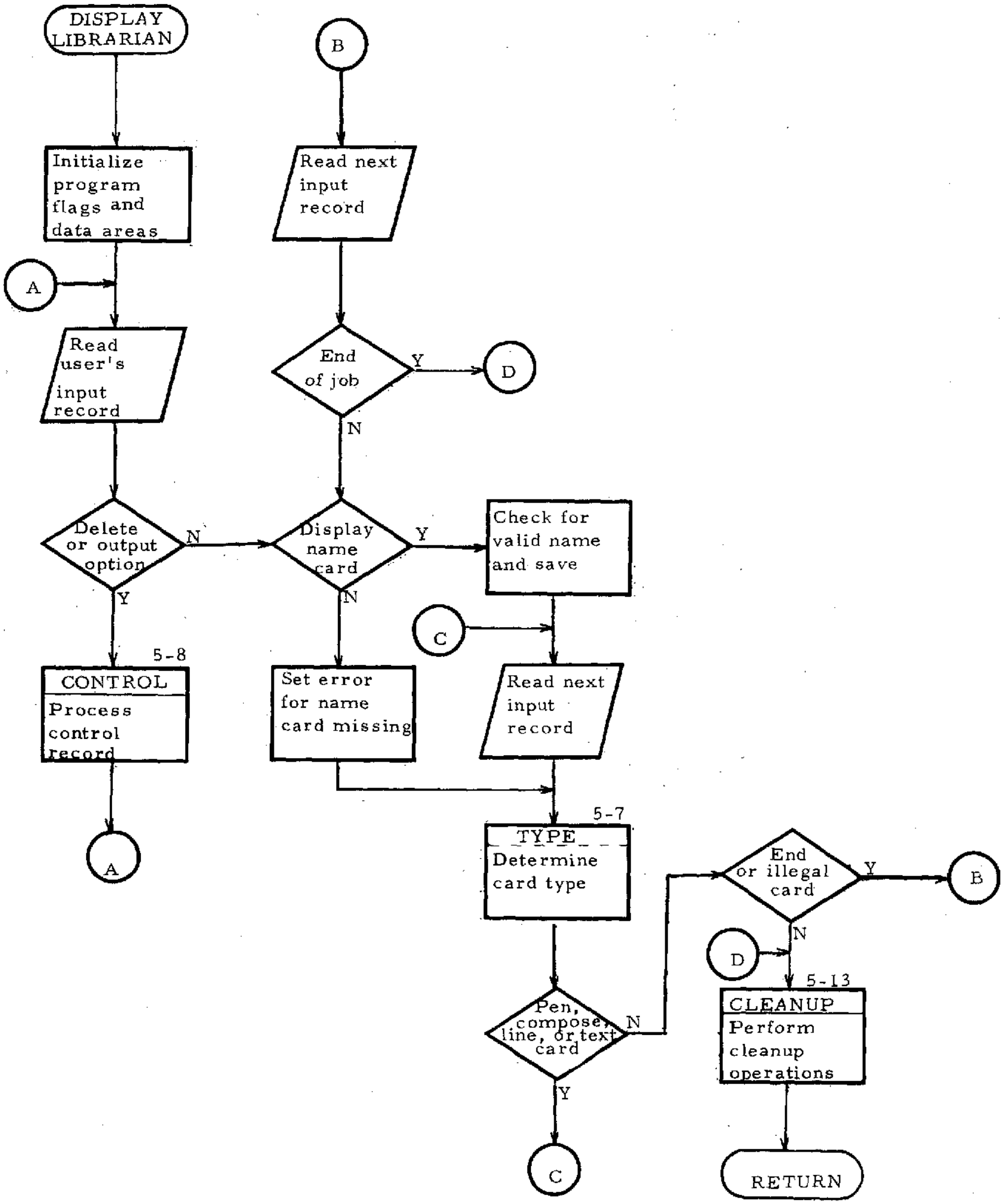

DISPLAY LIBRARIAN
B
Initialize program flags and data areas
Read next input record
A
Read user's input record
End of job
Y
D
N
Delete or output option
N
Display name card
Y
Check for valid name and save
Y
N
C
5-8
CONTROL
Process control record
Set error for name card missing
Read next input record
A
5-7
TYPE
Determine card type
End or illegal card
Y
B
N
D
5-13
Pen, compose, line, or text card
N
CLEANUP
Perform cleanup operations
Y
C
RETURN

Figure 5-6

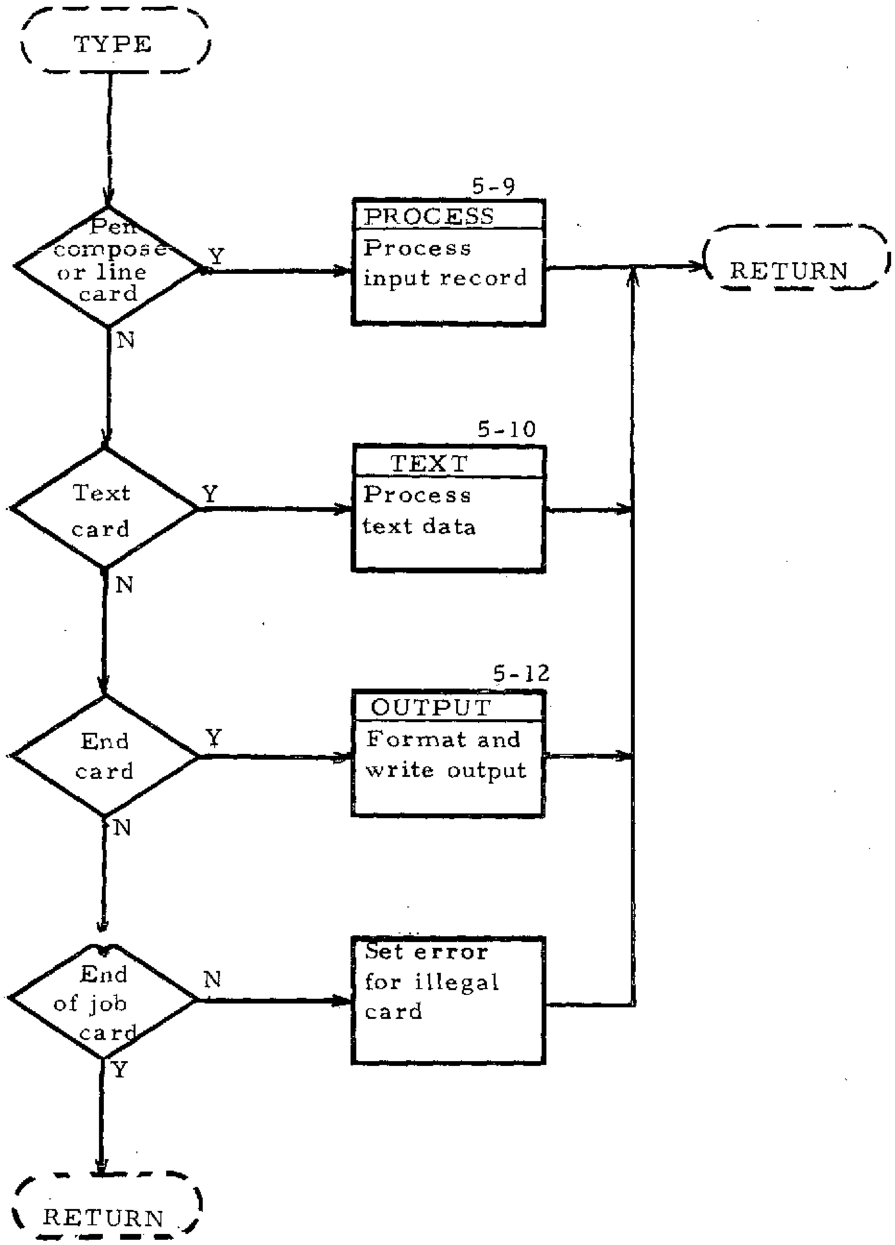

TYPE
5-9
PROCESS
Process input record
Pen compose or line card
Y
N
RETURN
5-10
TEXT
Process text data
Text card
Y
N
5-12
OUTPUT
Format and write output
End card
Y
N
End of job card
N
Y
Set error for illegal card
RETURN

Figure 5-7

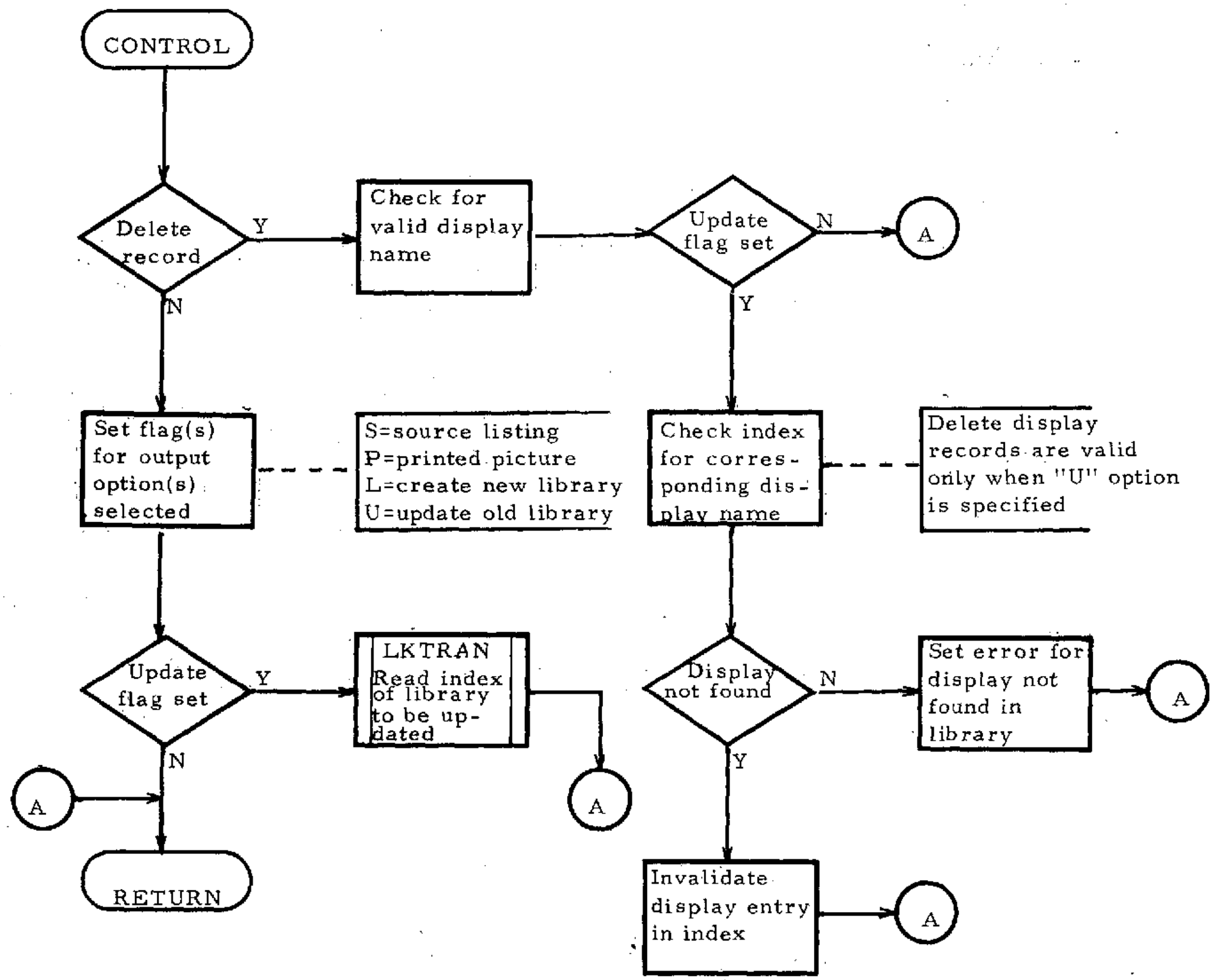

CONTROL
Delete record
Y
N
Check for valid display name
Update flag set
N
Y
A
Set flag(s) for output option(s) selected
S=source listing
P=printed picture
L=create new library
U=update old library
Check index for corresponding display name
Delete display records are valid only when "U" option is specified
Update flag set
Y
N
LKTRAN
Read index of library to be updated
A
A
RETURN
Display not found
N
Y
Set error for display not found in library
A
Invalidate display entry in index
A

Figure 5-8

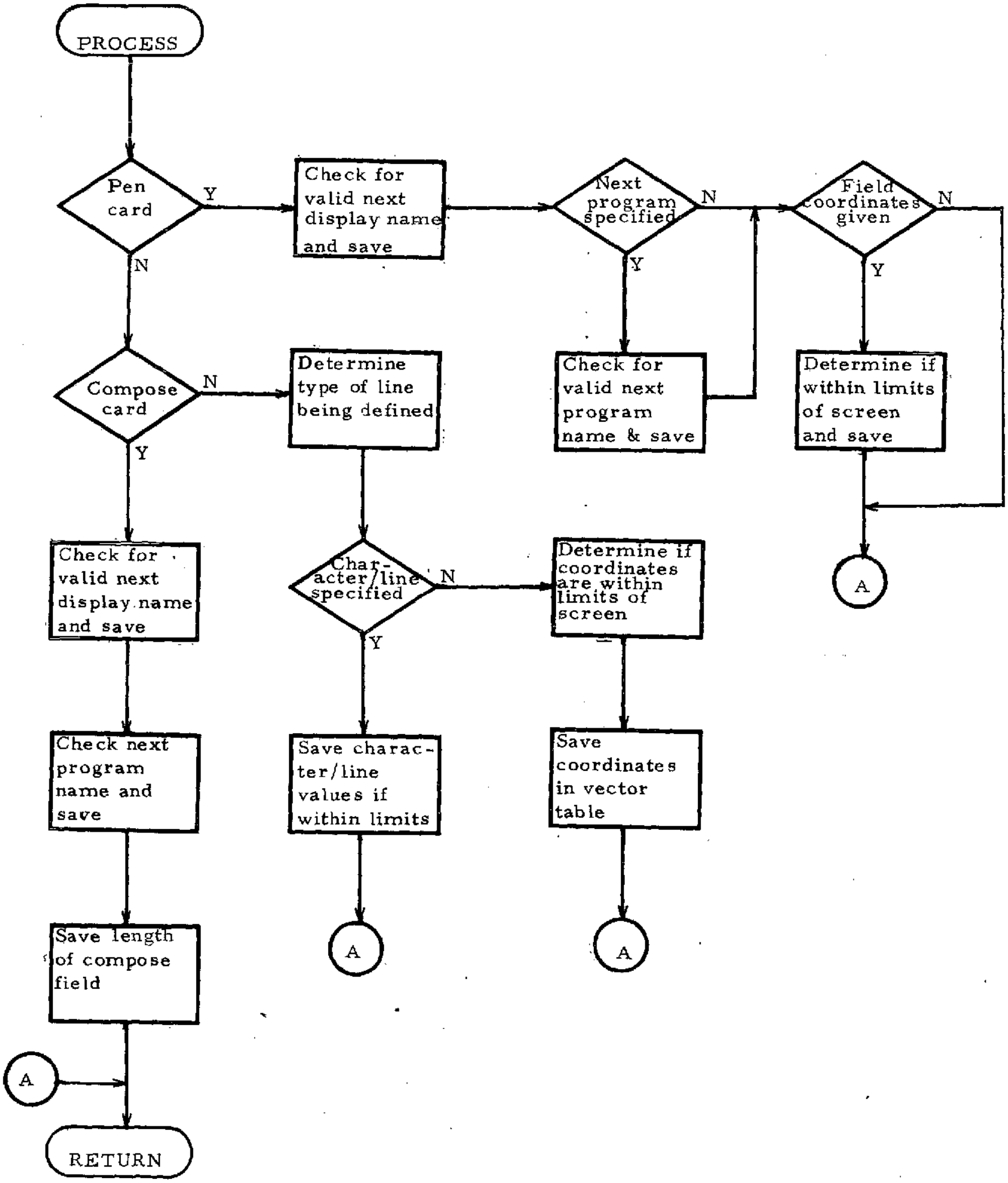

PROCESS
Pen card
Y
N
Check for valid next display name and save
Next program specified
N
Y
Check for valid next program name & save
Field coordinates given
N
Y
Determine if within limits of screen and save
A
Compose card
N
Y
Determine type of line being defined
Char-acter/line specified
N
Y
Determine if coordinates are within limits of screen
Save coordinates in vector table
A
Save charac-ter/line values if within limits
A
Check for valid next display name and save
Check next program name and save
Save length of compose field
A
RETURN

Figure 5-9

field entries are completed during the text processing (level 3) when the pen field characters "<" and ">" are encountered in the display text.

On receiving a compose card, the librarian saves the next display name, the next program name, and the compose field length in the keyboard page. The remaining information is supplied to the compose field entry during the text processing.

If the input record received, is a line card, the program determines the type of line being defined and compares the values specified against the limits allowed. If there is no error, it enters the specified values into either a coordinate table or a character/line table. Values entered in the coordinate tablet are not operated on again until the fourth level of processing when they are placed in the display text page in the form of vector commands. The values placed in the character/line table are converted to X-, Y-coordinate positions during text processing when the specified character number and line number are encountered.

5.2.3 Text Card Processing

During the text card processing (Figures 5-10 and 5-11), the librarian program performs three functions. These include verifying that the users text data can be successfully generated on the display screen completing the pen, key, and fill page entries and converting the vector entries in the character/line table to X-, Y-coordinates.

In determining whether the text can be generated on the screen, the librarian verifies that each text input record does not contain more characters than can fit on a single display line (i.e., 74 size 1 characters, 37 size 2 characters etc.), checks each character on the line to insure that it is displayable and verifies that the number of text lines does not exceed 35 (size 1). The pen field defining characters "<" and ">" and the change character size specifications are not included in the text character count.

On detecting a compose or fill-in character in the text, the Display Librarian stores the X-, Y-coordinates of the field in the key or fill page along with the present character size and, for fill field, the number of characters in the field. For compose fields the number of compose characters contained on the input record is compared against the value specified on the compose card to insure that there is no error in the field definition.

When pen field characters are encountered in the text, the X-, Y-coordinates are saved in the pen page along with the calculated ΔX and ΔY to the end of the field.

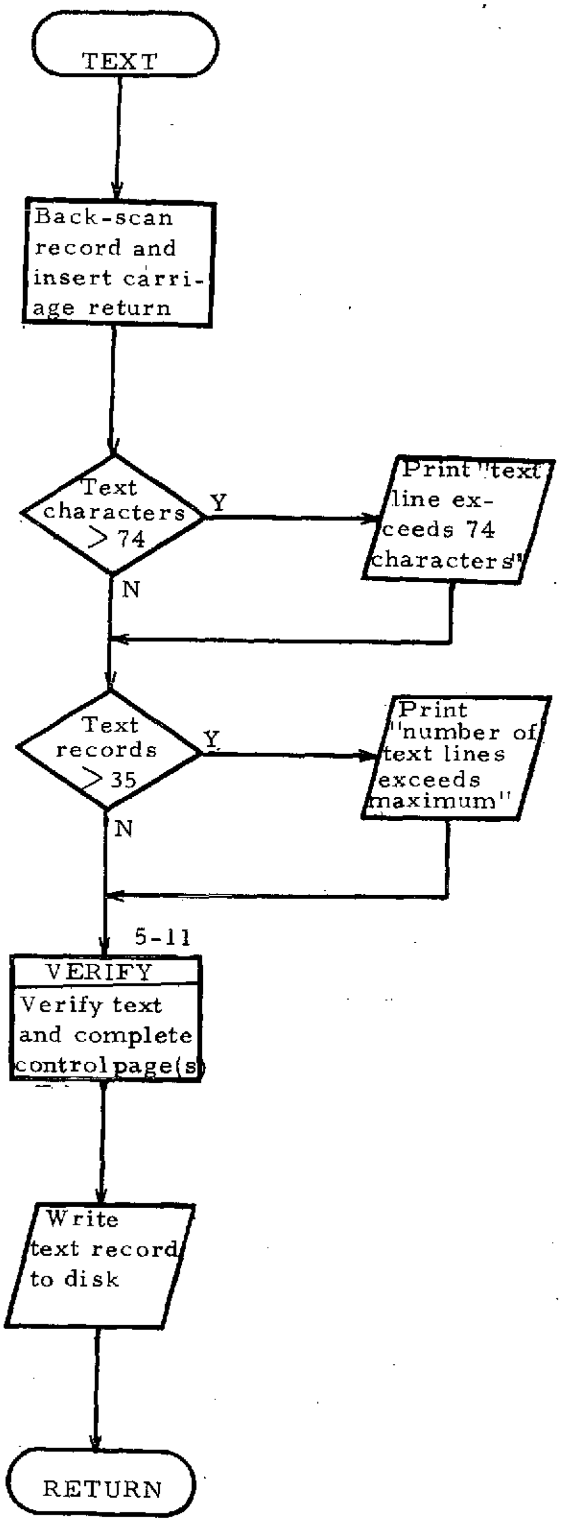

TEXT
Back-scan record and insert carriage return
Text characters > 74
Y
Print "text line exceeds 74 characters"
N
Text records > 35
Y
Print "number of text lines exceeds maximum"
N
5-11
VERIFY
Verify text and complete control page(s)
Write text record to disk
RETURN

Figure 5-10

VERIFY TEXT RECORD CONTENTS

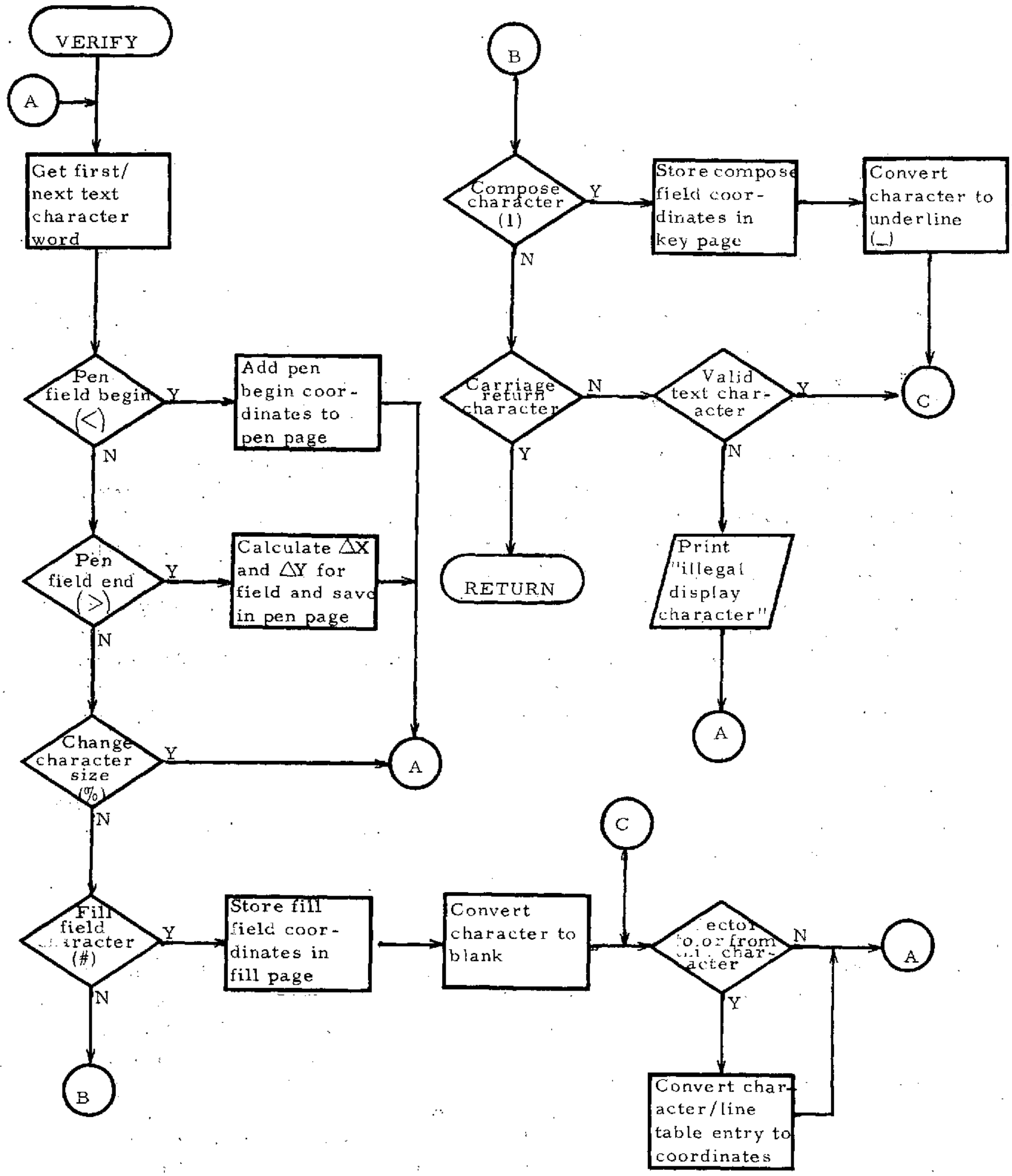


Figure 5-11

While the text characters are being "verified," the Librarian determines from the character/line table if a vector initiation or termination is specified for the present character/line position. If so, the character/line specification is replaced in the table, with the actual X-, Y-coordinates.

After each text record is read and verified, it is written to a scratch file on disk to be later operated upon by the output formatting routines.

5.2.4 Output Formatting

Upon receiving a display End record, the librarian enters the fourth level of display processing (Figure 5-12). It is on this level that the printed picture of the display is generated; if there are no errors in the display definition, the preformatted text pages are created and added to the display book along with the control pages.

Each text record for the display is retrieved from the librarian scratch file, formatted to effectively represent how the data will appear on the screen, and printed on the user assigned output device. If there are no errors, the display data is stored in the text page along with the necessary control commands to display on the screen. After all the text records have been retrieved, preformatted in the text pages, and written to the printer, the librarian uses the X-, Y-coordinates from both the coordinate table and character/line table and creates the display control commands to cause the generation of the specified vectors. These commands are stored following the user defined text data, thus completing the preformatted text page.

Upon completion of the text page, the Display Librarian adds the newly defined display to the display book on disk and updates the index to reflect its presence.

5.2.5 Cleanup

When attempting to read the next user input record the Display Librarian receives an end-of-job indication, this final level of processing is executed (Figure 5-13). If a new display book was created from the user's display data or an old book was updated without any of the old displays being deleted or replaced, the librarian adds the display index to the book on disk and terminates. If, however, during update of a display book, one or more of the old displays was deleted or replaced, the librarian at this time compresses the display chapters and the display index to eliminate all unused areas and then terminates.

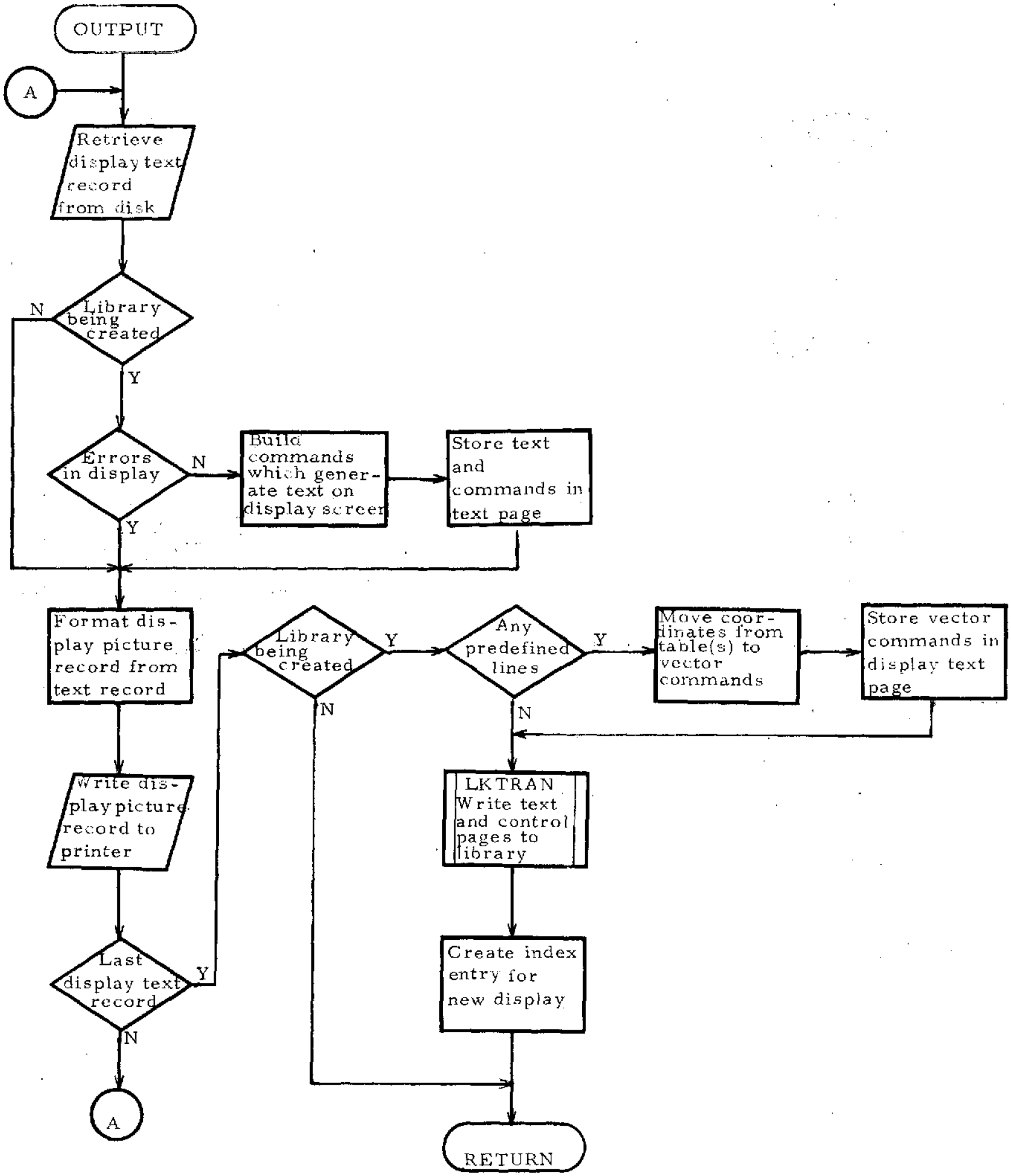


Figure 5-12

ORIGINAL PAGE IS OF POOR QUALITY

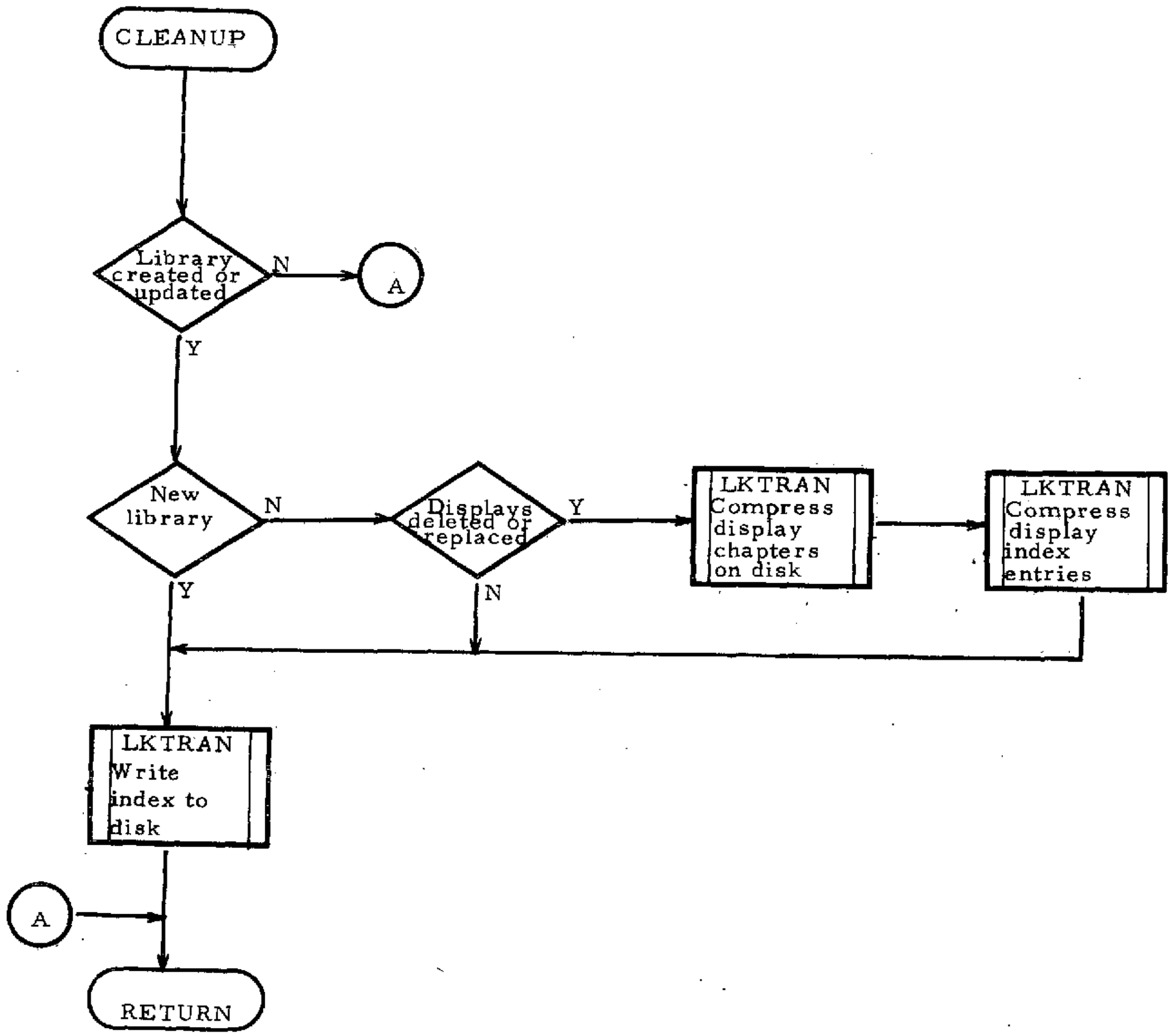

CLEANUP
Library created or updated
N
A
Y
New library
N
Displays deleted or replaced
Y
LKTRAN Compress display chapters on disk
LKTRAN Compress display index entries
Y
N
LKTRAN Write index to disk
A
RETURN

Figure 5-13

6. PROGRAM DISPLAYS AND OUTPUT

6.1 Program Displays

The user interacts with VIDS and controls the VIDS operation by responding to a set of displays which are placed on the screen. These displays are:

- o Mode Selection (see Figure 6-1)
- o System Parameter Selection (see Figure 6-2)
- o System Parameter Selection (Post-Analysis) (see Figure 6-3)
- o Aircraft Selection (see Figure 6-4)
- o Aircraft Dependent Parameters Selection (see Figure 6-5)
- o Aircraft Dependent Parameters Selection (Post-Analysis) (see Figure 6-6)

6.2 Program Output

Depending on which options are selected from the above displays, data will be output in one or a combination of the following methods:

- o Unprocessed data recorded on magnetic tape.
- o Display of vortex information in tabular form (see Figure 6-7).
- o Display of vortex positions as a function of time (see Figure 6-8).
- o Display of vortex locations in an X-Y coordinate system (see Figure 6-9).
- o Display of raw data with vortex centers marked (Scatter Plots) (see Figure 6-10).
- o Listing of raw data.

6.2.1 Magnetic Tape Output

Data is recorded on the magnetic tape in the following format:

- o One file per flyby.
- o One or more fixed length records per frame (1 record for each 1000 data points or portion of 1000) of LDV data.

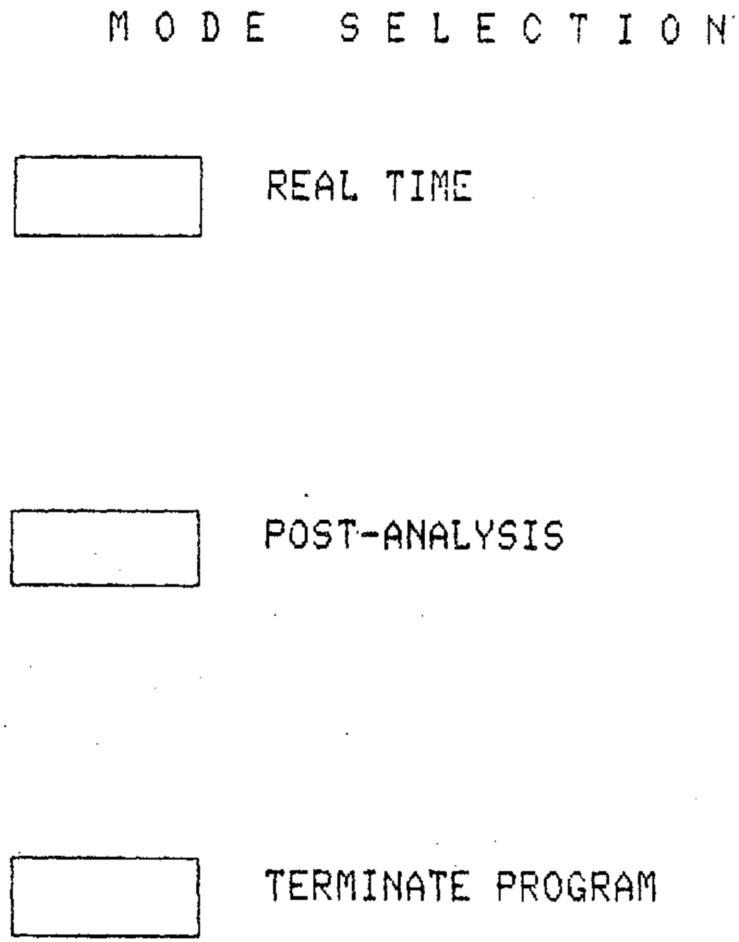


Figure 6-1

SYSTEM PARAMETER SELECTION

S Y S T E M   P A R A M E T E R S

INITIALIZATION

DAY ___

FLY-BY NO. _____

	0	1	2	3	4	5	NONE
ACQ TIME LIMIT(MIN)							

DATA SELECTION

	SELECTED	INHIBITED
VAN 1		
VAN 2		
MAG TAPE RECORDING		

DISPLAY SELECTION

	PRIMARY	ALTERNATE
X-Y PLOTS		
TIME BASED PLOTS		
TABULAR DATA		
AUTO HARD COPY	YES	NO

OPERATION

CONTINUOUS   PARAMETER SELECTION

Figure 6.3

S Y S T E M   P A R A M E T E R S
(POST-ANALYSIS)

FLY-BY SELECTION

DAY ___

FLY-BY NO. _____

TAPE REWIND [ ]

AUTO PROCESSING [ ]

VAN SELECTION

	SELECTED	INHIBITED
VAN 1	[ ]	[ ]
VAN 2	[ ]	[ ]

DISPLAY SELECTION

	PRIMARY	ALTERNATE
X-Y PLOTS	[ ]	[ ]
TIME-BASED PLOTS	[ ]	[ ]
TABULAR DATA	[ ]	[ ]
SCATTER PLOTS	[ ]	[ ]
AUTO HARD COPY	[ ] YES	[ ] NO

OPERATION

[ ] START FLY-BY   [ ] SELECT A/C DEPENDENT PARAMETERS

Figure 6-3

AIRCRAFT SELECTION

AIRCRAFT TYPE

@ B-707	@ L-1011	@ 2 ENGINE JET
@ B-727	@ C-880 (990)	@ 4 ENGINE JET
@ B-737	@ VC-10	@ 2 ENGINE PROP
@ B-747	@ BAC 111	@ 1 ENGINE PROP
@ DC-08	@ A 300	@ OTHER
@ DC-09	@ ILYUSHIN	@ ______
@ DC-10	@ YAC 40	

DEFAULT PLANE TYPE (POST ANALYSIS)

RETURN TO SYSTEM PARAMETER SELECTION

Figure 6-4

AIRCRAFT DEPENDENT PARAMETER SELECTION

FLY-BY NO. DAY TIME A/C

AIRCRAFT DEPENDENT PARAMETERS

CORRELATION CIRCLE

R = RADIUS (FT) 0 10 20 30 40 50 60 70 80 90 100

NOISE SPIKE FILTER

A=% V-PEAK NOISE TOLERANCE 0 10 20 30 40 50 60 70 80 90 100

B=% PTS IN TOLERANCE 0 10 20 30 40 50 60 70 80 90 100

NS= VORTEX SEPARATION 0 1 2 3 4 5 6 7 8 9 10

MISSING VORTEX CRITERIA

C=% POINTS FOR 2ND VORTEX 0 10 20 30 40 50 60 70 80 90 100

D=% V-PEAK FOR NEXT FRAME 0 10 20 30 40 50 60 70 80 90 100

START FLY-BY (OPERATOR CONTROLLED)

START FLY-BY (CONTINUOUS)

UPDATE DEFAULTS

RETURN TO AIRCRAFT SELECTION

Figure 6-5

AIRCRAFT DEPENDENT PARAMETER SELECTION (POST-ANALYSIS)

FLY-BY NO. DAY TIME A/C

AIRCRAFT DEPENDENT PARAMETERS
(POST ANALYSIS)

CORRELATION CIRCLE

R = RADIUS (FT) 0 10 20 30 40 50 60 70 80 90 100

NOISE SPIKE FILTER

A=% V-PEAK NOISE TOLERANCE 0 10 20 30 40 50 60 70 80 90 100

B=% PTS IN TOLERANCE 0 10 20 30 40 50 60 70 80 90 100

NS= VORTEX SEPARATION 0 1 2 3 4 5 6 7 8 9 10

LOST VORTEX CRITERIA

C=% POINTS FOR 2ND VORTEX 0 10 20 30 40 50 60 70 80 90 100

D=% V-PEAK FOR NEXT FRAME 0 10 20 30 40 50 60 70 80 90 100

START FLY-BY

UPDATE DEFAULTS

RETURN TO SYSTEM PARAMETER

Figure 6-6

DISPLAY OF VORTEX INFORMATION

FLY-BY 00029 DAY 310 TIME 00:06:37 A/C B-707

R= 49 A= 50 B= 50 NS= 02 C= 25 D= 50

FR	DATA PTS	COR P	PT S	NOISE P	NOISE S	ANGLE MN	ANGLE MX	PK P	VEL S	TIME	PORT POS X	PORT POS Y	STARB POS X	STARB POS Y
01	0015	008	003	00	00	14	16	039	037	000 2	-036	136	074	137
02	0030	016	000	00	02	13	15	046	000	002 2	-053	091		
03	0034	020	008	02	00	08	20	045	026	004 6	-050	076	033	076
04	0030	019	004	00	00	07	12	041	036	006 9	-061	072	067	067
05	0037	019	011	00	00	05	15	039	037	009.9	-071	072	066	256
06	0022	008	007	00	00	06	12	033	034	011.9	-058	062	055	258
07	0036	015	019	00	00	04	12	034	241	015.3	-094	061	083	248
08	0046	022	013	00	21	03	14	034	236	016.2	-107	062	103	242
09	0048	030	009	00	00	03	11	039	036	022.3	-118	046	083	239
10	0036	023	005	00	00	03	14	037	036	021.7	-121	252	134	036
11	0030	017	000	00	00	08	13	039	000	025 1	-129	252		
12	0026	025	000	00	00	05	15	043	000	026 8	-142	252		
13	0024	023	000	00	00	09	14	041	000	030 0	-151	350		
14	0031	031	000	00	00	08	16	043	000	031 9	-163	350		
15	0041	034	000	00	00	04	13	045	002	035 1	-181	042		
16	0040	037	002	00	00	03	18	039	002	036 8	-187	047		
17	0036	033	002	00	00	24	19	039	000	039 6	-206	045		
18	0042	037	002	00	00	03	20	039	000	042.1	-213	051		
19	0021	021	000	00	00	13	17	036	000	044.7	-220	043		
20	0030	030	000	00	00	13	25	037	200	047.3	-227	056		
21	0013	013	000	00	00	20	27	037	200	048.9	-232	067		
22	0008	008	000	00	00	22	26	037	200	052.9	-237	075		

FR	DATA PTS	COR P	PT S	NOISE P	NOISE S	ANGLE MN	ANGLE MX	PK P	VEL S	TIME	PORT POS X	PORT POS Y	STARB POS X	STARB POS Y
01	0010	004	025	00	00	11	18	028	025	000.8	-047	106	055	097
02	0010	002	004	00	00	09	13	025	030	003.2	-060	092	045	082
03	0023	009	227	03	03	05	20	030	025	005.6	-049	063	066	056
04	2022	004	314	21	00	06	14	030	027	007.6	-099	064	053	071
05	2024	004	010	00	00	03	12	021	028	011.1	-084	065	044	060
06	0017	007	008	00	00	06	12	027	028	012.8	-101	065	065	062
07	0014	003	011	00	00	05	10	021	028	016.1	-121	056	054	044
08	0012	003	008	00	00	03	09	023	028	017.2	-094	053	067	035
09	0013	004	006	00	00	04	09	023	028	021.1	-115	038	081	030
10	0014	003	011	00	00	03	08	023	027	022 0	-126	047	083	030
11	0313	002	005	00	02	03	09	030	032	026 0	-189	048	099	033
12	0008	001	007	00	00	03	10	019	027	027 1	-223	043	106	037
13	0006	002	004	00	00	03	06	019	025	031 4	-185	038	123	029
14	0011	004	006	00	00	03	07	030	030	031.6	-153	038	119	029
15	0006	002	222	00	00	03	08	027	023	035.8	-215	042	151	032
16	0005	000	225	02	00	06	08	000	027	036.9			152	034
17	0009	003	026	00	00	03	12	027	021	040.4	-284	046	160	026
18	0006	000	306	00	20	07	09	000	028	041.9			168	036
25	0035	000		00		03	06	014						

EOF READ

Figure 6-7

DISPLAY OF VORTEX POSITION

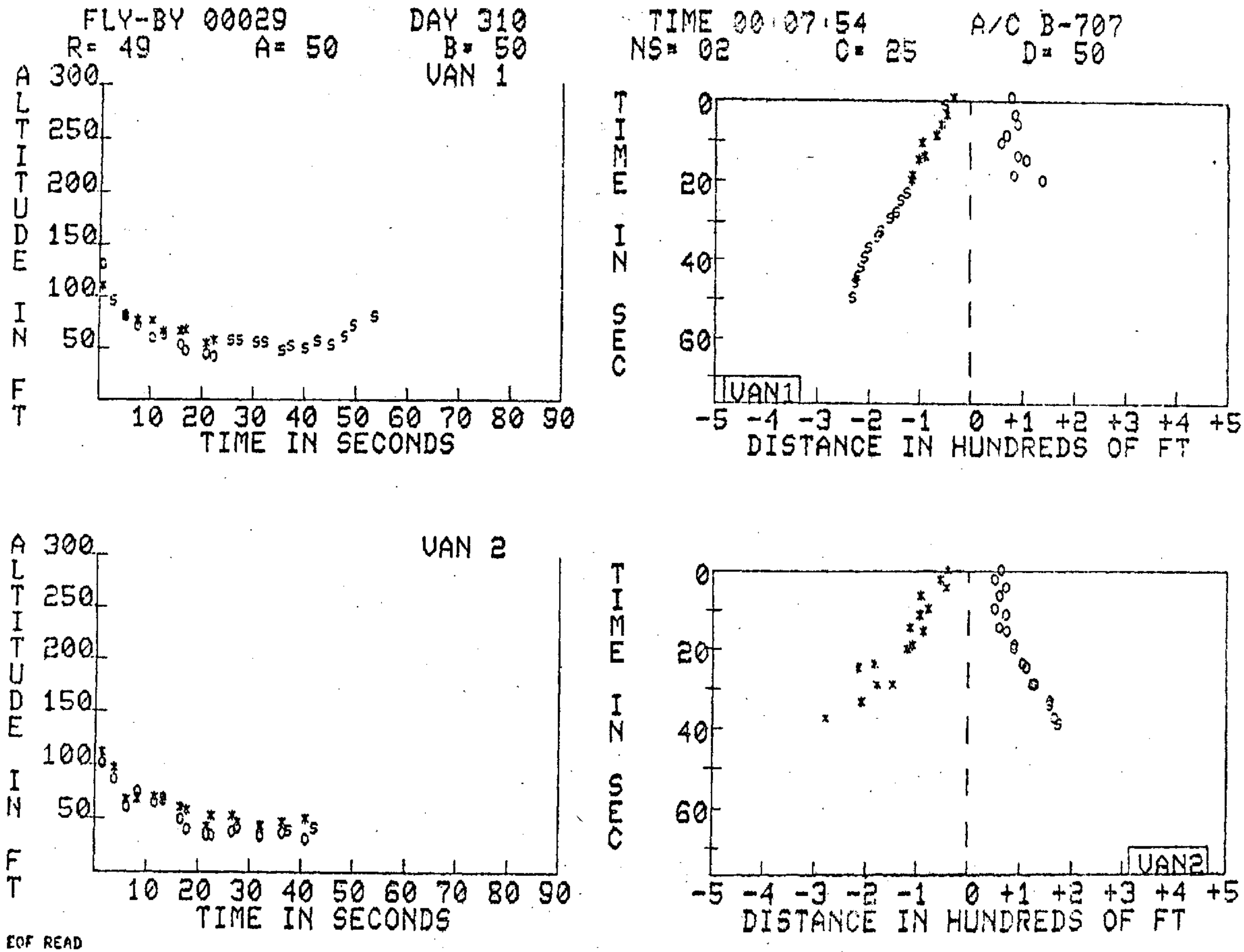


Figure 6-8

DISPLAY OF VORTEX LOCATION

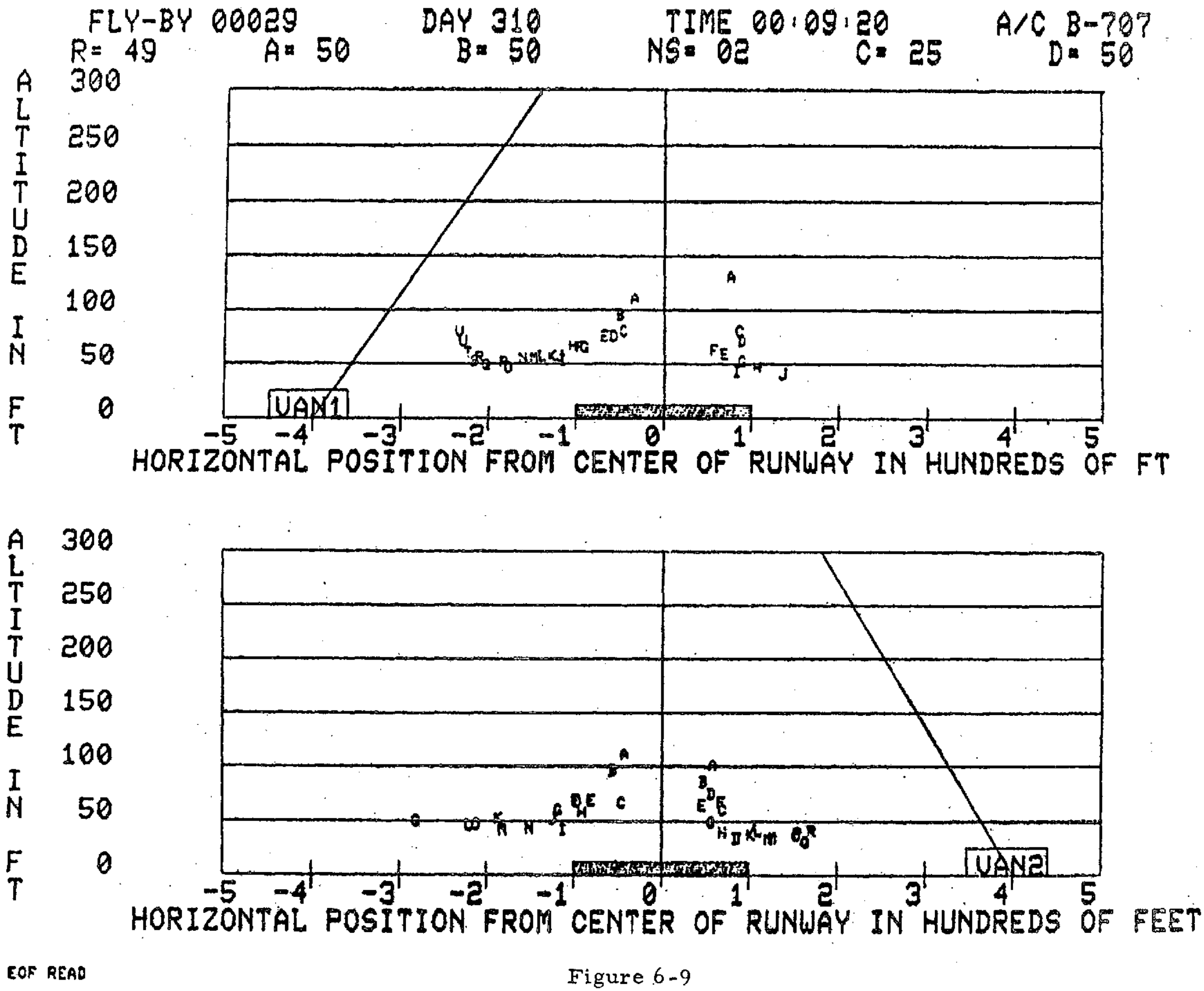


Figure 6-9

SAMPLE SCATTER PLOT

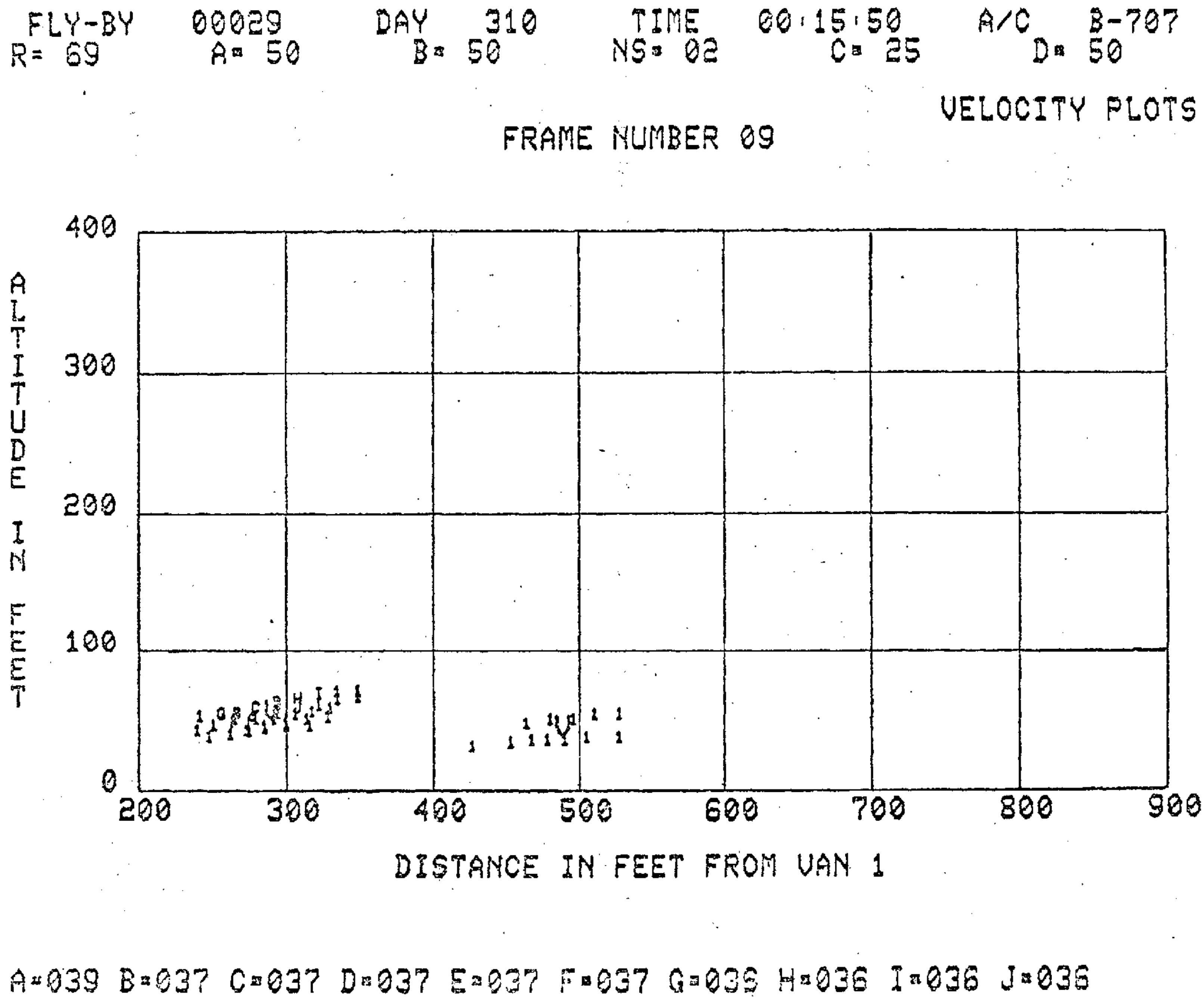


Figure 6-10

- Data from each LDV system is contained in separate records.
- There is no fixed order of records corresponding to a given LDV system.
- Each record contains 4009 integer words defined as follows:
  - IFLY - flyby number.
  - IFRM - frame number. A positive value indicates that the data was received from VAN 1. A negative value indicates that the data was received from VAN 2.
  - ITMINT (2) - two words containing the time that the first data point was received for this frame.
  - ITMEND (2) - two words containing the time that the last data point was received for this frame.
  - IDAY - number of the day of the year.
  - IPLN - index for plane type.
  - NUMPTS - number of data points in a frame.
  - IX - 1000 word array containing the X-coordinates for the data points in counts.
  - IY - 1000 word array containing the Y-coordinates for the data points in counts.
  - INTENS - 1000 word array containing intensity and filter information for the data points in counts. Bits 0 through 6 contain the number of filters and bits 7 through 14 contain the intensity of the point.
  - IVEL - 1000 word array containing velocity information for the data points in counts. Bits 0 through 6 contain the maximum velocity. Bits 8 through 14 contain the peak velocity.

6.2.2 Time-Based Plots

Time-based plots (see Figure 6-8) are generated according to the following format:

- The screen is divided into four quadriles.
- VAN 1 data is displayed in the top quadriles.

- VAN 2 data is displayed in the bottom quadriles.
- Height of the vortex centroid as a function of time is displayed in the left quadriles.
- Horizontal location of the vortex centroid as a function of time is displayed in the right quadriles.
- The maximum time that may be displayed is 90 seconds.
- An "*" is used to represent the port vortex.
- An "o" is used to represent the starboard vortex.
- An "s" is used to represent a single vortex.

6.2.3 X-Y Plots

X-Y plots (see Figure 6-9) are generated according to the following format:

- VAN 1 data is displayed on the top half of the screen.
- VAN 2 data is displayed on the bottom half of the screen.
- The vortex centroid is located in an X-Y coordinate system with frame 1 data represented by an A, frame 2 data by a B, etc., for the first 26 frames. After 26 frames the cycle is repeated.

6.2.4 Tabular Data

Tabular data (see Figure 6-7) is displayed according to the following format:

- VAN 1 data is displayed on the left half of the screen.
- VAN 2 data is displayed on the right half of the screen.
- A frame of data is entered as a blank line if there are 0 data points or more than 1000 data points.

The headings of the columns are:

- FR - 2-digit frame number. If the frame number is greater than 99, only the right 2 digits of the number are displayed.
- DATA PTS - number of points contained in this frame of data.

- COR PT - number of data points in the correlation area for a vortex. P represents port and S represents starboard.

- NOISE - the number of noise spikes which were found in processing the vortex information. P represents the number found while processing the port vortex and S represents the number found while processing the starboard vortex.

- ANGLE - MN represents the minimum angle at which a data point was found during the scan. MX represents the maximum angle at which a data point was found during the scan.

- PK VEL - P represents the peak velocity found in the port vortex. S represents the peak velocity found in the staboard vortex.

- TIME - represents the time at whichthe vortex centroid was found with respect to the time that the first data point was received from this flyby.

- PORT POS - gives the X and Y centroid values for the port vortex.

- STARB POS - gives the X and Y centroid values for the starboard vortex.

6.2.5 Scatter Plots

Scatter plots (see Figure 6-10) display raw data points showing their location in an X-Y coordinate system and their velocities within ten ft./sec. and also show the vortex centroid. The points are represented by numerals 0-9 with the following velocity correspondence:

-0 20-30 ft./sec.

-1 30-40 ft./sec.

-2 40-50 ft./sec.

-3 50-60 ft./sec.

-4 60-70 ft./sec.

-5 70-80 ft./sec.

-6 80-90 ft./sec.

-7 90-100 ft./sec.

-8 100-110 ft./sec.

-9 110-120 ft./sec.

\- The points having the ten highest values are represented by the corresponding first ten letters of the alphabet and their values are printed on the bottom of the display.

The centroids are represented by two large V's.

APPENDIX A

ATVWS LISTINGS

TABLE OF CONTENTS

```
        .NLIST  TTM
        .TITLE  ATVWS
        .MCALL  .PARAM,.EXIT,.WAITR
        .GLOBL  LOOPZ
        .GLOBL  DISPIO
        .GLOBL  LOCK
        .GLOBL  FTOR,INIT1,DISAB1
        .GLOBL  FIRSTA
        .GLOBL  STORES
        .GLOBL  WADR,FWRIT,NEXT,SECNO,LSECT
        .GLOBL  BUF
        .GLOBL  BUFSAT,BUFST,FRAMES,SFLAGA
        .GLOBL  INTAB,INTAA
        .GLOBL  HEADR
        .GLOBL  FILN,BUF1
        .GLOBL  ATVWS,HALT,BUSY
        .GLOBL  INITDS,LKTRAN,LNKBLK
        .GLOBL  INTAW,INTBW
        .GLOBL  INTBA,INTBB
        .GLOBL  DISABL,IFLD
        .GLOBL  FCNTR,FCNTW
        .GLOBL  LRSEC
        .GLOBL  INIT
        .GLOBL  FURST
        .GLOBL  STORA,STORB
        .GLOBL  INITBF,FIN,SAW,WRITE
000000  .PARAM
;*
        .MACRO  PUSH            ;SAVE REGISTERS
        MOV     R0,-(SP)
        MOV     R1,-(SP)
        MOV     R2,-(SP)
        MOV     R3,-(SP)
        MOV     R4,-(SP)
        MOV     R5,-(SP)
        .ENDM
;*
        .MACRO  POP             ;RESTORE REGISTERS
        MOV     (SP)+,R5
        MOV     (SP)+,R4
        MOV     (SP)+,R3
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        MOV     (SP)+,R0
        .ENDM
```

```
                        ;*
                        ;*      DR-11 DEFINITIONS
                        ;*
        167772                  OBUFA=167772            ;DR-11 A OUTPUT BUFFER
        167770                  SREGA=167770            ;DR-11 A STATUS REGISTER
        000300                  VECTA= 300              ;INTERRUPT VECTOR FOR DR-11 A
        000300                  LEVLA= 300              ;INTERRUPT LEVEL FOR DR-11 A
                        ;*
        167762                  OBUFB=167762            ;DR-11 B OUTPUT BUFFER
        167760                  SREGB=167760            ;DR-11 B STATUS REGISTER
        000310                  VECTB= 310              ;INTERRUPT VECTOR FOR DR-11 B
        000300                  LEVLB= 300              ;INTERRUPT LEVEL FOR DR-11 B
                        ;*
        167754                  IBUFC = 167754          ;DR-11 C INPUT BUFFER
        167752                  OBUFC = 167752          ;DR-11 C OUTPUT BUFFER
        167750                  SREGC = 167750          ;DR-11 C STATUS REGISTER
        000320                  VECTC = 320             ;INTERRUPT VECTOR FOR DR-11 C
        000300                  LEVLC = 300             ;INTERRUPT LEVEL (6) FOR DR-11 C
                        ;*
        000100                  LEVS  = 100
        000140                  IWRD  = 140             ;ENABLE A AND B INTERRUPTS
        000100                  IWRDC= 100              ;ENABLE A INTERRUPT DR-11 C
        000000                  CNT   = 0               ;WORD COUNT
        000002                  BNBT  = 2               ;BLOCK CONTROL WORD
        000200                  BB    = 200             ;BUFFER BUSY
        000304                  DISARA = 304
        000314                  DISARB = 314
```

```
000000                              ATVWS:
                                    ;*
                                    ;*      SET UP INTERRUPT VECTORS FOR DR-11A AND B
                                    ;*
000000  012700  000300                      MOV     #VECTA,R0       ;INITIALIZE INTERRUPT VECTOR A
000004  012720  000000G                     MOV     #INTAA,(R0)+    ;LOCATION FOR REQUEST A, DR-11 A
000010  012720  000300                      MOV     #LEVLA,(R0)+    ;PSW
000014  012720  000000G                     MOV     #INTAW,(R0)+    ;LOCATION FOR REQUEST B, DR-11 A
000020  012720  000300                      MOV     #LEVLA,(R0)+    ;PSW
                                    ;*
                                    ;*
000024  012700  000310                      MOV     #VECTB,R0       ;INTERRUPT VECTOR B
000030  012720  000000G                     MOV     #INTAB,(R0)+    ;LOCATION FOR REQUEST A, DR-11 B
000034  012720  000300                      MOV     #LEVLB,(R0)+    ;PSW
000040  012720  000000G                     MOV     #INTBW,(R0)+    ;LOCATION FOR REQUEST B, DR-11 B
000044  012720  000300                      MOV     #LEVLB,(R0)+    ;PSW
                                    ;*
000050  012700  000320                      MOV     #VECTC,R0       ;INITIALIZE INTERRUPT VECTOR
000054  012720  000136'                     MOV     #INTCA,(R0)+    ;REQUEST A, DR-11 C
000060  012720  000300                      MOV     #LEVLC,(R0)+    ;PSW (LEVEL 6)
000064  012720  000166'                     MOV     #INTCB,(R0)+    ;REQUEST B, DR-11 C
000070  012710  000300                      MOV     #LEVLC,(R0)     ;PSW (LEVEL 6)
                                    ;*
000074  004567  000000G                     JSR     R5,INITDS
000100  000000                      CNTLC:  .WORD   0
000102  005767  177772                      TST     CNTLC           ;ERROR
000106  001032                              BNE     ATV1            ;YES
000110  004567  000000G                     JSR     R5,LKTRAN
000114  000000G                             .WORD   FILN
000116  000556'                             .WORD   BLOCK
000120  000000G                             .WORD   BUF1
000122  000004                              .WORD   4               ;READ
000124  000000                      CNTLB:  .WORD   0
000126  005767  177772                      TST     CNTLB
000132  001020                              BNE     ATV1
                                    ;*
000134  000205                              RTS     R5              ;RETURN TO DISPLAY CONTROLLER
                                    ;*
000136                              INTCA:
000136  004567  000320                      JSR     R5,DISABL       ;INHIBIT DATA FOR THIS FLYBY
000142  016767  167754' 000002'             MOV     IBUFC,KPLN      ;READ PLANE ID
000150  016767  000002' 000000G             MOV     KPLN,HEADR       ;SET PLANE ID IN BLOCK HEADER
000156  012767  000001  000000'             MOV     #1,KSTFLY       ;SET START FLAG (0=WAIT, 1=START)
000164  000002                              RTI                      ;RETURN
                                    ;*
000166                              INTCB:
000166  000240                              NOP
000170  000240                              NOP
000172  000002                              RTI
                                    ;*
                                    ;*
000174                              ATV1:
000174  012700  000010                      MOV     #10,R0
000200  000167  000012                      JMP     HLT1
                                    ;*
000204                              HALT:
000204  004567  000252                      JSR     R5,DISABL       ;GO STOP INPUT
```

```
000210  012767  000100  177776'          MOV    #LEVS,PSW           ;GO TO LEVEL ONE
                                 ;*
000216                           HLT1:
000216  016067  001046' 000046           MOV    MSGAD(R0),MCS1
000224  012767  000002  000740           MOV    #2,MTWO
000232  004567  000000G                  JSR    R5,DISPIO           ;GO TO LARGE CHARACTERS
000236  000403                           BR     HLT5
000240  001166'                          .WORD  MCODE4
000242  001172'                          .WORD  MTWO
000244  001174'                          .WORD  MBIG
000246                           HLT5:
000246  004567  000000G                  JSR    R5,DISPIO           ;BLANK SCREEN
000252  000402                           BR     HLT4
000254  001164'                          .WORD  MCODE3
000256  001170'                          .WORD  MBLK
000260                           HLT4:
000260  004567  000000G                  JSR    R5,DISPIO           ;OUTPUT ERROR MESSAGE
000264  000403                           BR     HLT2
000266  001160'                          .WORD  MCODE1
000270  001200'                          .WORD  MLGTH1
000272  001060'                  MCS1:   .WORD  MSG1
                                 ;*
000274                           HLT2:
                                 ;*
000274  004567  000000G                  JSR    R5,DISPIO           ;REQUEST DISPLAY 1
000300  000403                           BR     HLT3
000302  001162'                          .WORD  MCODE2
000304  001204'                          .WORD  MDISP
000306  001176'                          .WORD  MONE
000310                           HLT3:
000310  000167  000000G                  JMP    LOOPZ
```

```
                                        ;*
                                        ;*      INITIALIZE DR-11 FOR INPUT (A AND/OR B)
                                        ;*
000314                                  INIT:
000314                                          PUSH                        ;SAVE REGISTERS
000330  005067  000000G                         CLR     FCNTR               ;CLEAR READ FRAME COUNTER
000334  005067  000000G                         CLR     FCNTW               ;CLEAR WRITE FRAME COUNTER
000340  005067  000000G                         CLR     FIN                 ;RESET TEST OVER FLAG
000344  004767  000210                          JSR     PC,INITBF           ;INITIALIZE BUFFERS
000350  016701  000000G                         MOV     IFLD,R1
000354  001013                                  BNE     INIT3               ;NOT NORMAL, GO CHECK
000356  012767  000140  167770'                 MOV     #IWRD,SREGA         ;INITIALIZE DR-11 A
000364  005067  167772'                         CLR     OBUFA               ;RESET A
000370                                  INIT2:
000370  012767  000140  167760'                 MOV     #IWRD,SREGB         ;INITIALIZE DR-11 B
000376  005067  167762'                         CLR     OBUFB               ;RESET B
000402  000407                                  BR      INIT4               ;GO RESTORE REGISTERS AND EXIT
                                        ;*
000404                                  INIT3:
000404  005301                                  DEC     R1
000406  001770                                  BEQ     INIT2
000410  012767  000140  167770'                 MOV     #IWRD,SREGA         ;INITIALIZE DR-11 A CONLY
000416  005067  167772'                         CLR     OBUFA               ;RESET A
000422                                  INIT4:
000422                                          POP                         ;RESTORE REGISTERS AND EXIT
000436  000205                                  RTS     R5
                                        ;*
                                        ;*      INITIALIZE DR-11 C FOR INPUT
                                        ;*
000440                                  INIT1:
000440  016767  167754' 000002'                 MOV     IBUFC,KPLN
000446  005067  000000'                         CLR     KSTFLY
000452  012767  000100  167750'                 MOV     #IWRDC,SREGC        ;ENABLE A INTERRUPT FOR DR-11 C
000460  000205                                  RTS     R5
```

```
                                       .SBTTL   DISABLE INPUT
                               ;*
                               ;*      INHIBIT DATA INPUT
                               ;*
000462                         DISABL:
000462                                 PUSH                       ;SAVE REGISTERS
000476  012700  000304                 MOV      #DISARA,R0
000502  012710  000000G                MOV      #INTAW,(R0)       ;DISARM DR-11 A
000506  012700  000314                 MOV      #DISARB,R0
000512  012710  000000G                MOV      #INTBW,(R0)       ;DISARM DR-11 B
000516  005267  000000G                INC      FIN               ;SET TEST OVER FLAG
000522  005067  167770'                CLR      SREGA             ;CLEAR ENABLE DR-11 A
000526  005067  167760'                CLR      SREGB             ;CLEAR ENABLE DR-11 B
000532                                 POP                        ;RESTORE REGISTERS
000546  000205                         RTS      R5                ;RETURN
                               ;*
                               ;*      DISABLE INTERRUPT FOR DR-11  C
                               ;*
000550                         DISAB1:
000550  005067  167750'                CLR      SREGC
000554  000205                         RTS      R5
                               ;*
                               ;*
000556  000000                 BLOCK:  .WORD    0
```

```
                                    .SBTTL  INITIALIZE BUFFERS
                            ;*
                            ;*      INITIALIZE BUFFERS (R2=0,LDV1 R2=2,LDV2)
                            ;*
000560                      INITBF:
000560                              PUSH                    ;SAVE REGISTERS
000574 016701 000000G               MOV     BUF,R1          ;PICK UP BUFFER COUNT
000600 012702 000000G               MOV     #BUFST,R2       ;SET BUFFER START ADDRESSES
000604                      INITB1:
000604 012203                       MOV     (R2)+,R3
000606 005063 000002                CLR     BNBT(R3)        ;FREE ALL BUFFERS
000612 005063 000000                CLR     CNT(R3)         ;CLEAR WORD COUNT
000616 005301                       DEC     R1
000620 001371                       BNE     INITB1
                            ;*
000622 012702 000000G               MOV     #BUFST,R2       ;SET INITIAL BUFFER FOR LDV1
000626 011267 000000G               MOV     (R2),BUFSAT
000632 012203                       MOV     (R2)+,R3
000634 012763 000200 000002         MOV     #BB,BNBT(R3)
000642 011267 000002G               MOV     (R2),BUFSAT+2
000646 011203                       MOV     (R2),R3
000650 012763 000200 000002         MOV     #BB,BNBT(R3)    ;SET INITIAL BUFFER FOR LDV 2
000656 012703 000000G               MOV     #STORES,R3
000662 012367 000000G               MOV     (R3)+,STORA
000666 011367 000000G               MOV     (R3),STORB
000672 005067 000000G               CLR     SFLAGA
000676 005067 000000G               CLR     LOCK
000702 005067 000002G               CLR     SFLAGA+2
000706 005067 000000G               CLR     FTOR
000712 005067 000002G               CLR     FTOR+2
000716 005067 000000G               CLR     FIRSTA
000722 005067 000002G               CLR     FIRSTA+2
000726 012767 000001 000000G        MOV     #1,FRAMES
000734 012767 000001 000002G        MOV     #1,FRAMES+2
000742 005067 000000G               CLR     BUSY            ;CLEAR READ BUSY FLAG
000746 005067 000000G               CLR     WADR
000752 005067 000000G               CLR     FWRIT
000756 005067 000000G               CLR     NEXT
000762 012767 000001 000002G        MOV     #1,NEXT+2
000770 005067 000000G               CLR     FURST
000774 005067 000002G               CLR     FURST+2
001000 005067 000000G               CLR     SECNO
001004 005067 000000G               CLR     LSECT
001010 012767 000001 000002G        MOV     #1,LSECT+2
001016 005067 000000G               CLR     LRSEC
001022 012767 000001 000002G        MOV     #1,LRSEC+2
001030                              POP                     ;RESTORE REGISTERS
001044 000207                       RTS     PC              ;RETURN
                            ;*
                            ;*
001046 001060'              MSGAD:  .WORD   MSG1
001050 001060'                      .WORD   MSG1
001052 001100'                      .WORD   MSG2
001054 001120'                      .WORD   MSG3
001056 001140'                      .WORD   MSG4
                            ;*
                            ;*      ERROR MESSAGES
```

```
                                                      ;*
   001060        117         125         124          MSG1:      .ASCII      /OUT OF BUFFERS  /
   001063        040         117         106
   001066        040         102         125
   001071        106         106         105
   001074        122         123         040
   001077        040
   001100        104         111         123          MSG2:      .ASCII      /DISK ERROR WRITE/
   001103        113         040         105
   001106        122         122         117
   001111        122         040         127
   001114        122         111         124
   001117        105
   001120        104         111         123          MSG3:      .ASCII      /DISK ERROR READ /
   001123        113         040         105
   001126        122         122         117
   001131        122         040         122
   001134        105         101         104
   001137        040
   001140        111         116         111          MSG4:      .ASCII      /INITIALIZE FILE /
   001143        124         111         101
   001146        114         111         132
   001151        105         040         106
   001154        111         114         105
   001157        040
                                                      ;*
   001160     000003                                  MCODE1:    .WORD       3
   001162     000002                                  MCODE2:    .WORD       2
   001164     000007                                  MCODE3:    .WORD       7
   001166     000016                                  MCODE4:    .WORD       14.
   001170     000001                                  MBLK:      .WORD       1
   001172     000002                                  MTWO:      .WORD       2
   001174     034033                                  MBIG:      .WORD       34033
   001176     177777                                  MONE:      .WORD       -1
   001200     000020                                  MLGTH1:    .WORD       16.
   001202     000000                                  MNULL:     .WORD       0
   001204     000001                                  MDISP:     .WORD       1
              000000'                                            .CSECT      KSTFL
   000000     000000                                  KSTFLY:    .WORD       0
   000002     000000                                  KPLN:      .WORD       0
                                                      ;*
              000001'                                            .END
```

```
ATVWS                                    MACRO VR05-01A 01-JAN-72 00:05 PAGE 6+
SYMBOL TABLE

ATVWS    000000RG          ATV1     000174R          BB     = 000200
BLOCK    000556R           BNBT   = 000002           BUF    = ****** G
BUFSAT= ****** G           BUFST  = ****** G         BUF1   = ****** G
BUSY   = ****** G          CNT    = 000000           CNTLB    000124R
CNTLC    000100R           DISABL   000462RG         DISAB1   000550RG
DISARA= 000304             DISARB= 000314            DISPIO= ****** G
FCNTR  = ****** G          FCNTW  = ****** G         FILN   = ****** G
FIN    = ****** G          FIRSTA= ****** G          FRAMES= ****** G
FTOR   = ****** G          FURST  = ****** G         FWRIT  = ****** G
HALT     000204RG          HEADR  = ****** G         HLT1     000216R
HLT2     000274R           HLT3     000310R          HLT4     000260R
HLT5     000246R           IBUFC  = 167754           IFLD   = ****** G
INIT     000314RG          INITBF   000560RG         INITB1   000604R
INITDS= ****** G           INIT1    000440RG         INIT2    000370R
INIT3    000404R           INIT4    000422R          INTAA  = ****** G
INTAB  = ****** G          INTAW  = ****** G         INTBA  = ****** G
INTBB  = ****** G          INTBW  = ****** G         INTCA    000136R
INTCB    000166R           IWRD   = 000140           IWRDC  = 000100
KPLN     000002R     002   KSTFLY   000000R     002  LEVLA  = 000300
LEVLB  = 000300            LEVLC  = 000300           LEVS   = 000100
LKTRAN= ****** G           LNKBLK= ****** G          LOCK   = ****** G
LOOPZ  = ****** G          LRSEC  = ****** G         LSECT  = ****** G
MBIG     001174R           MBLK     001170R          MCODE1   001160R
MCODE2   001162R           MCODE3   001164R          MCODE4   001166R
MCS1     000272R           MDISP    001204R          MLGTH1   001200R
MNULL    001202R           MONE     001176R          MSGAD    001046R
MSG1     001060R           MSG2     001100R          MSG3     001120R
MSG4     001140R           MTWO     001172R          NEXT   = ****** G
OBUFA  = 167772            OBUFB  = 167762           OBUFC  = 167752
PC     =%000007            PSW    = 177776           R0     =%000000
R1     =%000001            R2     =%000002           R3     =%000003
R4     =%000004            R5     =%000005           R6     =%000006
R7     =%000007            SAW    = ****** G         SECNO  = ****** G
SFLAGA= ****** G           SP     =%000006           SREGA  = 167770
SREGB  = 167760            SREGC  = 167750           STORA  = ****** G
STORB  = ****** G          STORES= ****** G          SWR    = 177570
VECTA  = 000300            VECTB  = 000310           VECTC  = 000320
WADR   = ****** G          WRITE  = ****** G
. ABS.   000000      000
         001206      001
KSTFL    000004      002

ERRORS DETECTED:  0
FREE CORE:   11635. WORDS
ATVWS,L3<ATVWS.2
```

APPENDIX B

READ LISTINGS

PRECEDING PAGE BLANK NOT FILMED

TABLE OF CONTENTS

```
                .NLIST    TTM
                .TITLE    READ DATA FROM LDV 1 AND 2
                .GLOBL    FTOR
                .GLOBL    HEADR
                .GLOBL    FIRSTA
                .GLOBL    WADR,FWRIT,NEXT,SECND
                .GLOBL    FILN,FIN,WRITE
                .GLOBL    INTAA,INTAB,INTBA,INTBB
                .GLOBL    SAW
                .GLOBL    BUFSAT,BUFST,FRAMES,SFLAGA
                .GLOBL    INTAW,INTBW,BUSY
                .GLOBL    FURST
                .GLOBL    STORES
                .GLOBL    LKTRAN,REQUST
                .GLOBL    LNKBLK,FCNTW
                .GLOBL    LOCK
                .GLOBL    HALT
                .GLOBL    BUF
                .GLOBL    LSECT
                .GLOBL     STORA,STORB
                .MCALL    .WAITR
                .MCALL    .PARAM
000000          .PARAM
```

```
;*
        .MACRO  PUSH            ;SAVE REGISTERS
        MOV     R0,-(SP)
        MOV     R1,-(SP)
        MOV     R2,-(SP)
        MOV     R3,-(SP)
        MOV     R4,-(SP)
        MOV     R5,-(SP)
        .ENDM
;*
        .MACRO  POP             ;RESTORE REGISTERS
        MOV     (SP)+,R5
        MOV     (SP)+,R4
        MOV     (SP)+,R3
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        MOV     (SP)+,R0
        .ENDM
```

```
000340                LEVX = 340
000100                LEVS = 100            ;PROCESS LEVEL 2
000000                CNT   = 0             ;WORD COUNT
000002                BNBT  = 2             ;BLOCK CONTROL
000004                HDRW = 4              ;HEADER WORD
000006                FNNP  = 6             ;FRAME NUMBER,SECTOR NUMBER
000010                TIMS1 = 8.            ;START TIME, HIGH ORDER
000012                TIMS2 = 10.           ;START TIME, HIGH ORDER
000014                TIMR1 = 12.           ;READ TIME , LOW ORDER
000016                TIMR2 = 14.           ;READ TIME , HIGH ORDER
000040                EOF   = 40            ;END OF FRAME
000200                BB    = 200           ;BUFFER BUSY
000100                BW    = 100           ;BUFFER WAITING FOR WRITE
000020                EOT   = 20            ;END OF TEST
000076                DATNO = 62.           ;NO. OF DATA POINTS PER BUFFER
000010                SIZB  = 8.
000370                SIZD  = 248.          ;DATA WORDS PER BLOCK
              ;*
167774                IBUFA=167774          ;DR-11 A INPUT BUFFER
000304                VECTA = 304           ;B REQUEST VECTOR ADDRESS LDV 1
167764                IBUFB=167764          ;DR-11 B INPUT BUFFER
000314                VECTB = 314           ;B REQUEST VECTOR ADDRESS LDV 2
006246                TIMX2 = 6246
006244                TIMX1 = 6244
```

```
                                               .SBTTL   REQUEST A(END OF FRAME)
                                       ;*
                                       ;*      REQUEST A(END OF FRAME) INTERRUPT  DR-11 A
                                       ;*
 000000                                INTAA:
 000000  012767  000340  177776'               MOV      #LEVX,PSW          ;GO TO LEVEL 7
 000006                                        PUSH                        ;SAVE REGISTERS
 000022  012702  000004                        MOV      #4,R2              ;SET DATA WORD COUNTER
 000026                                INTAA1:
 000026  016703  167774'                       MOV      IBUFA,R3           ;BRING IN DATA AND DISCARD
 000032  005302                                DEC      R2
 000034  001374                                BNE      INTAA1
 000036  005767  023442                        TST      SFLAGA             ;FIRST EOF
 000042  001014                                BNE      NO3
 000044  005267  023434                        INC      SFLAGA             ;SET FLAG=NOT FIRST FRAME
 000050  012767  000276' 000304'               MOV      #INTBA,VECTA       ;SET DATA READ ROUTINE LDV 1
 000056                                        POP
 000072  000002                                RTI                         ;FIRST EOF INTERRUPT LDV 1
 000074                                NO3:
 000074  005062  023510'                       CLR      FIRSTA(R2)         ;RESET FIRST DATA FLAG
 000100  016201  023514'                       MOV      BUFSAT(R2),R1      ;PICK UP POINTER TO CURRENT BUFFER
 000104  052761  000040  000002                BIS      #EOF,BNBT(R1)      ;SET END OF FRAME INDICATOR
 000112  004767  000444                        JSR      PC,SAW             ;SWITCH BUFFERS
 000116  005262  023500'                       INC      FRAMES(R2)
                                       ;*
 000122  000167  000304                        JMP      INTBA3            ;GO CHECK FOR WRITE READY
                                       ;*
                                       ;*      REQUEST A (END OF FRAME) INTERRUPT DR-11 B
                                       ;*
 000126                                INTAB:
 000126  012767  000340  177776'               MOV      #LEVX,PSW
 000134                                        PUSH                        ;SAVE REGISTERS
 000150  012702  000004                        MOV      #4,R2              ;SET DATA WORD COUNTER
 000154                                INTAB1:
 000154  016703  167764'                       MOV      IBUFB,R3           ;BRING IN DATA AND DISCARD
 000160  005302                                DEC      R2
 000162  001374                                BNE      INTAB1
 000164  005202                                INC      R2                 ;SET INDIACTOR FOR LDV2
 000166  005202                                INC      R2
 000170  005767  023312                        TST      SFLAGB             ;FIRST EOF
 000174  001337                                BNE      NO3                ;NO
 000176  012767  000512' 000314'               MOV      #INTBB,VECTB       ;SET DATA READ ROUTINE LDV 2
 000204  005267  023276                        INC      SFLAGB             ;SET FLAG=NOT FIRST FRAME
 000210                                        POP
 000224  000002                                RTI                         ;FIRST EOF INTERRUPT LDV 2
```

```
                                                .SBTTL     READ DATA FROM DR-11
                                        ;*
                                        ;*      IPUT DATA AND DISCARD
                                        ;*
000226                                  INTAW:
000226  012767  000004  023276                  MOV     #4,CNTRA            ;SET WORD COUNT
000234                                  INTAW1:
000234  016767  167774' 023266                  MOV     IBUFA,DATA          ;BRING IN DATA WORDS AND DISCARD
000242  005367  023264                          DEC     CNTRA
000246  001372                                  BNE     INTAW1
000250  000002                                  RTI                         ;EXIT
                                        ;*
                                        ;*
000252                                  INTBW:
000252  012767  000004  023254                  MOV     #4,CNTRB
000260                                  INTBW1:
000260  016767  167764' 023242                  MOV     IBUFB,DATA          ;BRING IN DATA AND DISCARD
000266  005367  023242                          DEC     CNTRB
000272  001372                                  BNE     INTBW1
000274  000002                                  RTI                         ;EXIT
                                        ;*
                                        ;*
                                        ;*              READ DATA FROM DR-11 A
                                        ;*
000276                                  INTBA:
000276  012767  000340  177776'                 MOV     #LEVX,PSW
000304                                          PUSH                        ;SAVE REGISTER (R0-R4)
000320  016701  001014                          MOV     STORA,R1            ;PICK UP STORE ADDRESS POINTER
000324  012702  000004                          MOV     #4,R2               ;SET WORD COUNTER N
000330                                  INTBA1:
000330  016721  167774'                         MOV     IBUFA,(R1)+         ;BRING IN DATA WORD N
000334  005302                                  DEC     R2                  ;N=N+1
000336  001374                                  BNE     INTBA1              ;N=0, FINISHED
000340                                  INTBA4:
000340  010162  001340'                         MOV      R1,STORA(R2)
000344  016201  023514'                         MOV     BUFSAT(R2),R1
000350  005762  023510'                         TST     FIRSTA(R2)          ;TEST FOR FIRST DATA
000354  001010                                  BNE     INTBA5              ;NOT FIRST
000356  016761  006244' 000010                  MOV     TIMX1,TIMS1(R1)     ;SET START TIME
000364  016761  006246' 000012                  MOV     TIMX2,TIMS2(R1)
000372  005262  023510'                         INC     FIRSTA(R2)
000376                                  INTBA5:
000376  016761  006244' 000014                  MOV     TIMX1,TIMR1(R1)     ;SAVE TIME OF DATA READ
000404  016761  006246' 000016                  MOV     TIMX2,TIMR2(R1)
000412  005272  023514'                         INC     @BUFSAT(R2)         ;INCREMENT WORD COUNT
000416  027227  023514' 000076                  CMP     @BUFSAT(R2),#DATNO  ;BUFFER FULL
000424  103402                                  BLO     INTBA3              ;NO, GO EXIT
                                        ;*
000426  004767  000130                          JSR     PC,SAW              ;GO SWITCH BUFFERS AND WRITE FULL
000432                                  INTBA3:
000432  005767  023036                          TST     REQUST              ;ANY REQUESRS FOR WRITES
000436  001416                                  BEQ     INTBA7              ;NO, GO EXIT
                                        ;*
000440  005767  023014                          TST     BUSY
000444  100413                                  BMI     INTBA7
000446  005767  000664                          TST     LOCK
```

```
000454  012767  000002  000654              MOV     #2,LOCK
000462  012767  000100  177776'             MOV     #LEVS,PSW
000470  004767  000242                      JSR     PC,WRITE            ;GO CHECK ON WRITE READY
                                    ;*
000474                              INTBA7:
000474                                      POP                         ;RESTORE REGISTERS (R0-R4)
000510  000002                              RTI                         ;EXIT
                                    ;*
                                    ;*              READ DATA FROM DR-11 B
                                    ;*
000512                              INTBB:
000512  012767  000340  177776'             MOV     #LEVX,PSW
000520                                      PUSH                        ;SAVE REGISTERS (R0-R4)
000534  016701  000602                      MOV     STORB,R1            ;PICK UP STORE ADDRESS POINTER
000540  012702  000004                      MOV     #4,R2               ;SET WORD COUNT
000544                              INTBB1:
000544  016721  167764'                     MOV     IBUFB,(R1)+         ;BRING IN DATA WORD N
000550  005302                              DEC     R2                  ;N=N+1
000552  001374                              BNE     INTBB1              ;N=0, FINISHED
000554  005202                              INC     R2                  ;YES, SET DR-11 B INDICATOR IN R2
000556  005202                              INC     R2
000560  000667                              BR      INTBA4              ; GO SWITCH BUFFERS AND WRITE
                                    ;*
```

```
                                         .SBTTL   SWITCH AND WRITE
                                  ;*
                                  ;*     SWITCH BUFFERS AND WRITE FULL ONE TO DISK
                                  ;*              R2 = 0, LDV 1
                                  ;*              R2 = 2, LDV 2
                                  ;*
000562                            SAW:
000562 010200                            MOV      R2,R0              ;SAVE R2
000564 016705 000664                     MOV      BUF,R5             ;PICK UP NUMBER OF BUFFERS
000570 012701 001344'                    MOV      #BUFST,R1          ;PICK UP BUFFER STARTS
000574 012702 001410'                    MOV      #STORES,R2         ;PICK UP BUFFER STORES
000600                            SAW1:
000600 012103                            MOV      (R1)+,R3           ;SET BUFFER START
000602 012204                            MOV      (R2)+,R4           ;SET BUFFER STORE
000604 036327 000002 000200              BIT      BNBT(R3),#BB       ;BUFFER BUSY
000612 001406                            BEQ      SAW2               ;NO, GO SWITCH
000614 005305                            DEC      R5                 ;YES,GO TO NEXT
000616 001370                            BNE      SAW1
                                  ;*
000620                            SAW4:
000620 012700 000002                     MOV      #2,R0              ;PICK UP ERROR CODE
000624 000167 000000G                    JMP      HALT
000630                            SAW2:
000630 016005 023514'                    MOV      BUFSAT(R0),R5      ;SAVE OLD (FULL) BUFFER
000634 010360 023514'                    MOV      R3,BUFSAT(R0)      ;SET NEW BUFFER START
000640 012763 000200 000002              MOV      #BB,BNBT(R3)
000646 010460 001340'                    MOV      R4,STORA(R0)       ;SET NEW BUFFER STORE
                                  ;*
000652 004767 000436                     JSR      PC,CALBLK          ;GO GET NEXT SECTOR NUMBER
000656 010465 000006                     MOV      R4,FNNP(R5)        ;SET SECTOR NUMBER IN BLOCK
                                  ;*
                                  ;*
                                  ;*
000662 016765 022610 000004              MOV      HEADR,HDRW(R5)     ;SET HEADER WORD
000670 016004 023500'                    MOV      FRAMES(R0),R4      ;PICK UP FRAME NUMBER
000674 005700                            TST      R0                 ;LDV 2
000676 001401                            BEQ      SAW3               ;NO
000700 005404                            NEG      R4                 ;YES, NEGATE
000702                            SAW3:
000702 072427 000010                     ASH      #8.,R4             ;POSITION
000706 050465 000000                     BIS      R4,CNT(R5)         ;SET WITH WORD COUNT
                                  ;*
000712 052765 000100 000002              BIS      #BW,BNBT(R5)       ;SET WRITE READY BIT
000720 005267 022550                     INC      REQUST             ;SET WRITE REQUST
000724 012760 000002 023524'             MOV      #2,FURST(R0)
000732 010002                            MOV      R0,R2
                                  ;*
000734 000207                            RTS      PC
```

```
                                                .SBTTL  WRITE BUFFERS TO DISK
                                        ;*
                                        ;*      WRITE BUFFERS TO DISK
                                        ;*
000736                                  WRITE:
000736  010200                                  MOV     R2,R0
                                        ;*
000740                                  WRIT1:
000740  005767  022512                          TST     FWRIT           ;HAS WRITE BEEN REQUESTED
000744  001444                                  BEQ     WRIT10          ;NO
000746                                          .WAITR  #LNKBLK,#WRITS1
000760  005767  000264                          TST     CNTW            ;ERROR IN LAST WRITE
000764  001404                                  BEQ     WRIT5
                                        ;*
000766  012700  000004                          MOV     #4,R0           ;PICK UP ERROR CODE
000772  000167  000000G                         JMP     HALT
000776                                  WRIT5:
                                        ;*
000776  016701  000242                          MOV     WADR,R1         ;PICK UP LAST BUFFER ADDRESS
001002  016746  177776'                         MOV     PSW,-(SP)
001006  012767  000340  177776'                 MOV     #LEVX,PSW
001014  036127  000002  000100                  BIT     BNBT(R1),#BW
001022  001404                                  BEQ     WRITX
001024  005061  000000                          CLR     CNT(R1)
001030  005061  000002                          CLR     BNBT(R1)        ;CLEAR LAST BUFFER
001034                                  WRITX:
001034  012667  177776'                         MOV     (SP)+,PSW
001040  005000                                  CLR     R0              ;INITIALIZE LDV INDICATOR (1)
001042  036727  022424  000001                  BIT     SECND,#1        ;LAST WRITE FOR LDV 2
001050  001402                                  BEQ     WRIT10          ;NO
001052  012700  000002                          MOV     #2,R0           ;YES,START WITH THAT ONE
001056                                  WRIT10:
001056  012704  000002                          MOV     #2,R4           ;PICK UP LOOP COUNTER
001062                                  WRIT7:
001062  012702  001344'                         MOV     #BUFST,R2       ;SET START OF BUFFERS
001066  016701  000362                          MOV     BUF,R1          ;SET NUMBER OF BUFFERS
001072                                  WRIT3:
001072  012203                                  MOV     (R2)+,R3        ;TRY BUFFER  N
001074  036327  000002  000100                  BIT     BNBT(R3),#BW    ;BUFFER WAITING ON WRITE
001102  001015                                  BNE     WRIT4           ;YES
001104                                  WRIT6:
001104  005301                                  DEC     R1              ;NO, FINISHED THIS LOOP
001106  001371                                  BNE     WRIT3           ;NO, TRY NEXT ONE
                                        ;*
001110  005067  022342                          CLR     FWRIT
001114  162700  000002                          SUB     #2,R0           ;YES,SET LDV INDICATOR FOR NEXT LOOP
001120  001402                                  BEQ     WRIT8
001122  012700  000002                          MOV     #2,R0
                                        ;*
001126                                  WRIT8:
001126  005304                                  DEC     R4              ;ALL LOOPS FINISHED
001130  001354                                  BNE     WRIT7           ;NO, TRY NEXT
                                        ;*
001132  000167  000146                          JMP     WRITS1
                                        ;*
001136                                  WRIT4:
001136  005700                                  TST     R0              ;LDV 1 OR 2
```

```
001140  001405                           BEQ     WRIT9               ;LDV 1
                                 ;*
001142  005763  000000                   TST     CNT(R3)             ;LDV 2
001146  100356                           BPL     WRIT6               ;WRONG BUFFER
001150  000167  000006                   JMP     WRITS               ;BUFFER OK, GO WRITE IT
                                 ;*
001154                           WRIT9:
001154  005763  000000                   TST     CNT(R3)             ;RIGHT BUFFER
001160  100751                           BMI     WRIT6               ;WRONG ,GO   TRY AGAIN
                                 ;*
001162                           WRITS:
001162  016005  023462'                  MOV     NEXT(R0),R5         ;PICK UP LAST SECTOR WRITTEN
001166  001410                           BEQ     WRITSA              ;FIRST TIME
001170  005305                           DEC     R5
001172  001406                           BEQ     WRITSA              ;FIRST TIME FOR LDV 2
001174  005205                           INC     R5
001176  166305  000006                   SUB     FNNP(R3),R5         ;CHECK LAST SECTOR CALCULATED
001202  062705  000002                   ADD     #2,R5
001206  001336                           BNE     WRIT6               ;NOT NEXT IN SEQUENCE, TRY AGAIN
001210                           WRITSA:
001210  016360  000006  023462'          MOV     FNNP(R3),NEXT(R0)
001216  016367  000006  022246           MOV     FNNP(R3),SECNO      ;SET THIS SECTOR IN CALL SEQUENCE
001224  010367  000014                   MOV     R3,WADR             ;SET BUFFER ADDRESS IN CALL SEQUENCE
001230  005067  000014                   CLR     CNTW                ;CLEAR FUNCTION CODE
001234  004567  000000G                  JSR     R5,LKTRAN
001240  023536'                          .WORD   FILN
001242  023472'                          .WORD   SECNO
001244  000000                   WADR:   .WORD   0
001246  000002                           .WORD   2
001250  000000                   CNTW:   .WORD   0
001252  012767  000001  022176           MOV     #1,FWRIT
001260  036327  000002  000040           BIT     BNBT(R3),#EOF
001266  001404                           BEQ     WRITS2
001270  005260  000000G                  INC     FTOR(R0)
001274  005267  000000G                  INC     FCNTW
001300                           WRITS2:
001300  005367  022170                   DEC     REQUST              ;
001304                           WRITS1:
001304  010002                           MOV     R0,R2
001306  005067  000024                   CLR     LOCK
001312  000207                           RTS     PC                  ;RETURN
```

```
                                                .SBTTL    WRITE BLOCK NO.
                                       ;*
                                       ;*
                                       ;*       CALCULATE NEXT WRITE BLOCK
                                       ;*                 R0 = 0, LDV 1
                                       ;*                 R0 = 2, LDV 2
                                       ;*
 001314                                CALBLK:
 001314  005760  023524'                        TST       FURST(R0)
 001320  001403                                 BEQ       CAL1
 001322  062760  000002  023466'                ADD       #2,LSECT(R0)
 001330                                CAL1:
 001330  016004  023466'                        MOV       LSECT(R0),R4
 001334  000207                                 RTS       PC
```

```
                                .SBTTL  INPUT BUFFERS
                        ;*
                        ;*      DATA INPUT BUFFERS
001336  000000          LOCK:   .WORD   0               ;WRITE LOCK (0=FREE, 2=LOCKED)
001340                  STORE:
001340  000000          STORA:  .WORD   0               ;CURRENT STORE ADDRESS FOR LDV 1
001342  000000          STORB:  .WORD   0               ;CURRENT STORE ADDRESS FOR LDV 2
                        ;*
001344                  BUFST:                          ;BUFFER START ADDRESSES
001344  001456'                 .WORD   IEA1A
001346  002456'                 .WORD   IEA2A
001350  003456'                 .WORD   IEA3A
001352  004456'                 .WORD   IEA4A
001354  005456'                 .WORD   IEA1B
001356  006456'                 .WORD   IEA2B
001360  007456'                 .WORD   IEA3B
001362  010456'                 .WORD   IEA4B
001364  011456'                 .WORD   IEB1A
001366  012456'                 .WORD   IEB2A
001370  013456'                 .WORD   IEB3A
001372  014456'                 .WORD   IEB4A
001374  015456'                 .WORD   IEB1B
001376  016456'                 .WORD   IEB2B
001400  017476'                 .WORD   IEB3B
001402  020456'                 .WORD   IEB4B
001404  021456'                 .WORD   IEC1A
001406  022456'                 .WORD   IEC2A
                        ;*
001410                  STORES:                         ;BUFFER STORE ADDRESSES
001410  001476'                 .WORD   ISA1A
001412  002476'                         ISA2A
001414  003476'                         ISA3A
001416  004476'                         ISA4A
001420  005476'                         ISA1B
001422  006476'                         ISA2B
001424  007476'                         ISA3B
001426  010476'                         ISA4B
001430  011476'                         ISB1A
001432  012476'                         ISB2A
001434  013476'                         ISB3A
001436  014476'                         ISB4A
001440  015476'                         ISB1B
001442  016476'                         ISB2B
001444  017456'                         ISB3B
001446  020476'                         ISB4B
001450  021476'                 .WORD   ISC1A
001452  022476'                 .WORD   ISC2A
                        ;*
001454  000022          BUF:    .WORD   18.
                        ;*
001456                  IEA1A:  .BLKW   SIZB
001476                  ISA1A:  .BLKW   SIZD
                        ;*
002456                  IEA2A:  .BLKW   SIZB
002476                  ISA2A:  .BLKW   SIZD
```

```
003476          ISA3A:  .BLKW   SIZD
                ;*
004456          IEA4A:  .BLKW   SIZB
004476          ISA4A:  .BLKW   SIZD
                ;*
005456          IEA1B:  .BLKW   SIZB
005476          ISA1B:  .BLKW   SIZD
                ;*
006456          IEA2B:  .BLKW   SIZB
006476          ISA2B:  .BLKW   SIZD
                ;*
007456          IEA3B:  .BLKW   SIZB
007476          ISA3B:  .BLKW   SIZD
                ;*
010456          IEA4B:  .BLKW   SIZB
010476          ISA4B:  .BLKW   SIZD
                ;*
011456          IEB1A:  .BLKW   SIZB
011476          ISB1A:  .BLKW   SIZD
                ;*
012456          IEB2A:  .BLKW   SIZB
012476          ISB2A:  .BLKW   SIZD
                ;*
013456          IEB3A:  .BLKW   SIZB
013476          ISB3A:  .BLKW   SIZD
                ;*
014456          IEB4A:  .BLKW   SIZB
014476          ISB4A:  .BLKW   SIZD
                ;*
015456          IEB1B:  .BLKW   SIZB
015476          ISB1B:  .BLKW   SIZD
                ;*
016456          IEB2B:  .BLKW   SIZB
016476          ISB2B:  .BLKW   SIZD
                ;*
017456          ISB3B:  .BLKW   SIZB
017476          IEB3B:  .BLKW   SIZD
                ;*
020456          IEB4B:  .BLKW   SIZB
020476          ISB4B:  .BLKW   SIZD
                ;*
021456          IEC1A:  .BLKW   SIZB
021476          ISC1A:  .BLKW   SIZD
                ;*
022456          IEC2A:  .BLKW   SIZB
022476          ISC2A:  .BLKW   SIZD
```

```
                                                .SBTTL  CONSTANTS
 023456  000000                    FWRIT:       .WORD   0
 023460  000000                    BUSY:        .WORD   0               ;WRITE BUSY FLAG
 023462  000000  000001            NEXT:        .WORD   0,1             ;NEXT SECTOR FOR LDV 1 AND LDV 2
 023466  000000  000001            LSECT:       .WORD   0,1             ;LAST SECTOR FOR LDV 1 AND LDV 2
 023472  000000                    SECNO:       .WORD   0               ;LAST SECTOR WRITTEN
 023474  000000                    REQUST:      .WORD   0               ;WRITE REQUST COUNT
 023476  000000                    HEADR:       .WORD   0               ;HEADER (AIRCRAFT TYPE)
 023500                            FRAMES:
 023500  000000                    FRAMEA:      .WORD   0               ;FRAME COUNT FOR A
 023502  000000                    FRAMEB:      .WORD   0               ;FRAME COUNT FOR B
 023504  000000                    SFLAGA:      .WORD   0               ;START FLAG  A
 023506  000000                    SFLAGB:      .WORD   0               ;START FLAG  B
 023510  000000                    FIRSTA:      .WORD   0               ;FIRST DATA FLAG
 023512  000000                    FIRSTB:      .WORD   0               ;FIRST DATA FLAG
 023514                            BUFSAT:
                                                .BLKW   4
                                   ;*
 023524  000000  000000            FURST:       .WORD   0,0
 023530  000000                    DATA:        .WORD   0               ;DATA (DISCARDED)
 023532  000000                    CNTRA:       .WORD   0               ;WORD COUNTER A
 023534  000000                    CNTRB:       .WORD   0               ;WORD COUNTER B
                                   ;*
 023536                            FILN:
 023536  045666  073300                         .RAD50  /LDVS/          ;FILE NAME FOR LDVS
 023542  000000                                 .RAD50  //
 023544  000000                    FIN:         .WORD   0               ;FINISH FLAG
         000001'                                .END
```

SYMBOL TABLE

```
BB      = 000200         BNBT   = 000002       BUF       001454RG
BUFSAT    023514RG       BUFST    001344RG     BUSY      023460RG
BW      = 000100         CALBLK   001314R      CAL1      001330R
CNT     = 000000         CNTRA    023532R      CNTRB     023534R
CNTW      001250R        DATA     023530R      DATNO   = 000076
EOF     = 000040         EOT    = 000020       FCNTW   = ****** G
FILN      023536RG       FIN      023544RG     FIRSTA    023510RG
FIRSTB    023512R        FNNP   = 000006       FRAMEA    023500R
FRAMEB    023502R        FRAMES   023500RG     FTOR    = ****** G
FURST     023524RG       FWRIT    023456RG     HALT    = ****** G
HDRW    = 000004         HEADR    023476RG     IBUFA   = 167774
IBUFB   = 167764         IEA1A    001456R      IEA1B     005456R
IEA2A     002456R        IEA2B    006456R      IEA3A     003456R
IEA3B     007456R        IEA4A    004456R      IEA4B     010456R
IEB1A     011456R        IEB1B    015456R      IEB2A     012456R
IEB2B     016456R        IEB3A    013456R      IEB3B     017476R
IEB4A     014456R        IEB4B    020456R      IEC1A     021456R
IEC2A     022456R        INTAA    000000RG     INTAA1    000026R
INTAB     000126RG       INTAB1   000154R      INTAW     000226RG
INTAW1    000234R        INTBA    000276RG     INTBA1    000330R
INTBA3    000432R        INTBA4   000340R      INTBA5    000376R
INTBA7    000474R        INTBB    000512RG     INTBB1    000544R
INTBW     000252RG       INTBW1   000260R      ISA1A     001476R
ISA1B     005476R        ISA2A    002476R      ISA2B     006476R
ISA3A     003476R        ISA3B    007476R      ISA4A     004476R
ISA4B     010476R        ISB1A    011476R      ISB1B     015476R
ISB2A     012476R        ISB2B    016476R      ISB3A     013476R
ISB3B     017456R        ISB4A    014476R      ISB4B     020476R
ISC1A     021476R        ISC2A    022476R      LEVS    = 000100
LEVX    = 000340         LKTRAN= ****** G      LNKBLK= ****** G
LOCK      001336RG       LSECT    023466RG     NEXT      023462RG
NO3       000074R        PC     =%000007       PSW     = 177776
REQUST    023474RG       R0     =%000000       R1      =%000001
R2      =%000002         R3     =%000003       R4      =%000004
R5      =%000005         R6     =%000006       R7      =%000007
SAW       000562RG       SAW1     000600R      SAW2      000630R
SAW3      000702R        SAW4     000620R      SECNO     023472RG
SFLAGA    023504RG       SFLAGB   023506R      SIZB    = 000010
SIZD    = 000370         SP     =%000006       STORA     001340RG
STORB     001342RG       STORE    001340R      STORES    001410RG
SWR     = 177570         TIMR1  = 000014       TIMR2   = 000016
TIMS1   = 000010         TIMS2  = 000012       TIMX1   = 006244
TIMX2   = 006246         VECTA  = 000304       VECTB   = 000314
WADR      001244RG       WRITE    000736RG     WRITS     001162R
WRITSA    001210R        WRITS1   001304R      WRITS2    001300R
WRITX     001034R        WRIT1    000740R      WRIT10    001056R
WRIT3     001072R        WRIT4    001136R      WRIT5     000776R
WRIT6     001104R        WRIT7    001062R      WRIT8     001126R
WRIT9     001154R        .SYM   = 000027
. ABS.    000000      000
          023546      001

ERRORS DETECTED:  0
FREE CORE:  11556. WORDS
READ,L1<READ.1
```

APPENDIX C

FILL LISTINGS

TABLE OF CONTENTS

```
                    .NLIST     TTM
                    .TITLE     FILL PROCESS BUFFER
                    .GLOBL     FTOR
                    .GLOBL     NEXT
                    .GLOBL     FILL,BUSY,FIN,REQUST
                    .GLOBL     LKTRAN,LNKBLK,WRITE
                     .GLOBL     IFLD,LRSEC
                    .GLOBL     BUF1
                    .GLOBL     FCNTW,FCNTR,FILN
                    .MCALL     .PARAM,.WAITR
 000000             .PARAM
```

```
                .MACRO  PUSH            ;SAVE REGISTERS
                MOV     R0,-(SP)
                MOV     R1,-(SP)
                MOV     R2,-(SP)
                MOV     R3,-(SP)
                MOV     R4,-(SP)
                MOV     R5,-(SP)
                .ENDM
;*
                .MACRO  POP             ;RESTORE REGISTERS
                MOV     (SP)+,R5
                MOV     (SP)+,R4
                MOV     (SP)+,R3
                MOV     (SP)+,R2
                MOV     (SP)+,R1
                MOV     (SP)+,R0
                .ENDM
```

```
000000          ONE   = 0          ;START OF PROCESS BUFFER  X
003720          TWO   = 2000.      ;START OF Y
007640          THREE = 4000.      ;START OF  N/I
013560          FOUR  = 6000.      ;START OF V/V
001740          MPNTS = 992.       ;MAXIMUM NUMBER OF POINTE
000000          CNT   = 0          ;WORD COUNT
000002          BNBT  = 2          ;BLOCK CONTROL
000004          HDRW  = 4          ;HEADER WORD
000006          FMNP  = 6          ;FRAME NUMBER,SECTOR NUMBER
000010          TIMS1 = 8.         ;START TIME, LOW ORDER
000012          TIMS2 = 10.        ;START TIME, HIGH ORDER
000014          TIMR1 = 12.        ;READ TIME, LOW ORDER
000016          TIMR2 = 14.        ;READ TIME, HIGH ORDER
000200          BB = 200           ;BUFFER BUSY
000040          EOF = 40           ;END OF FRAME
000010          SIZB = 8.          ;WORDS PER CONTROL BLOCK
000370          SIZD = 248.        ;DATA WORDS PER BLOCK
000340          LEVW = 340
```

```
                                        .SBTTL  READ DATA FROM DISK
                                ;*
                                ;*      READ 1000 WORDS OR FRAME OF DATA
                                ;*
                                ;*      FORMAT FOR DATA PROCESSING PROGRAM
                                ;*
                                ;*
                                ;*
                                ;*      FILL DATA BUFFER
                                ;*
000000                          FILL:
000000                                  PUSH
000014                          FILL0:
000014  005067  000004'                 CLR     IEOFI
000020  026767  003076  003072          CMP     FCNTW,FCNTR     ;ALL FRAMES READ
000026  101020                          BHI     FILL1           ;NO, GO READ ONE
000030  005767  000000G                 TST     FIN             ;YES, TEST OVER
000034  001011                          BNE     FILLE           ;YES
000036  016701  000000'                 MOV     ABTRM,R1
000042  162701  000041                  SUB     #33.,R1
000046  001406                          BEQ     FILLE1
000050  162701  000011                  SUB     #9.,R1
000054  001403                          BEQ     FILLE1
000056  000756                          BR      FILL0           ;NO, WAIT FOR A FRAME
000060                          FILLE:
000060  005267  000004'                 INC     IEOFI           ;YES, SET END OF TEST FLAG
000064                          FILLE1:
000064  000167  000474                  JMP     FIXT
                                ;*
                                ;*
000070                          FILL1:
000070  016701  000000'                 MOV     IFLD,R1         ;PICK UP INHIBIT FLAG
000074  001404                          BEQ     SECT5           ;BOTH LDVS
000076  162701  000002                  SUB     #2,R1
000102  001422                          BEQ     SECT3           ;LDV 1 ONLY
000104  000416                          BR      SECT2           ;LDV 2 ONLY
000106                          SECT5:
000106  016701  002766                  MOV     FTOR,R1
000112  001413                          BEQ     SECT2
000114  016702  002762                  MOV     FTOR+2,R2
000120  001413                          BEQ     SECT3
000122  160102                          SUB     R1,R2
000124  100411                          BMI     SECT3
000126  001401                          BEQ     SECT6
000130  000404                          BR      SECT2
                                ;*
000132                          SECT6:
000132  036727  002746  000001          BIT     READST,#1
000140  001003                          BNE     SECT3
000142                          SECT2:
000142  012702  000002                  MOV     #2,R2
000146  000401                          BR      SECT4
                                ;*
000150                          SECT3:
000150  005002                          CLR     R2
                                ;*
000152                          SECT4:
```

```
000152  016267  003106' 002724          MOV     LRSEC(R2),READST
000160  010267  002726                  MOV     R2,REDEX
000164  016746  177776'                 MOV     PSW,-(SP)
000170  016767  000340' 177776'         MOV     LEVW,PSW
000176  012767  177777  000000G         MOV     #-1,BUSY
000204  012667  177776'                 MOV     (SP)+,PSW
000210  005067  002700                  CLR     FFG
000214  005067  002676                  CLR     FFLG
000220  005067  002700                  CLR     MAXW1
000224  005067  002676                  CLR     MAXW2
000230  005000                          CLR     R0              ;SET INDICATOR FOR FIRST READ
000232  004767  000544                  JSR     PC,READ         ;GO READ NUMBER 1
000236  016704  002666                  MOV     PRODAT,R4       ;SET STORE ADDRESS
000242  012701  000001                  MOV     #1,R1           ;SET MAX VELOCITY COUNTER
000246                          FILL2:
000246  012700  000002                  MOV     #2,R0           ;SET INDICATOR FUR SECOND READ
000252  004767  000524                  JSR     PC,READ         ;GO READ NUMBER 2
                                ;*
000256  016702  000612                  MOV     READRS,R2
000262  016203  000000                  MOV     CNT(R2),R3      ;SET DATA POINT COUNT
000266  042703  177400                  BIC     #177400,R3      ;CLEAR HIGH BYTE
000272  005703                          TST     R3
000274  001512                          BEQ     FILEOF
000276  016702  000566                  MOV     READ1,R2
000302                          FILL3:
                                
000302  004767  000304                  JSR     PC,STORE        ;SET 4 DATA WORDS
000306  004767  000322                  JSR     PC,MAXV         ;GO FIND MAX VELOCITY
000312  005724                          TST     (R4)+           ;INCREMENT TO NEXT PROCESS WORD
000314  005201                          INC     R1              ;INCREMENT TO NEXT MAX WORD
000316  005303                          DEC     R3              ;FINISHED BLOCK
000320  001370                          BNE     FILL3           ;NO
                                ;*
000322                          FILLB:
000322  012702  001100'                 MOV     #BUF1,R2
000326  004767  000366                  JSR     PC,TIFR
000332  005767  002560                  TST     FFLG            ;FIRST BLOCK
000336  001010                          BNE     FILLA           ;NO
000340  016267  000010  000004'         MOV     TIMS1(R2),TMINT
000346  016267  000012  000006'         MOV     TIMS2(R2),TMINT+2
000354  005367  002536                  DEC     FFLG            ;RESET FIRST FLAG
000360                          FILLA:
000360  036227  000002  000040          BIT     BNBT(R2),#EOF   ;EOF IN BLOCK
000366  001055                          BNE     FILEOF          ;YES
000370  020127  001740                  CMP     R1,#MPNTS       ;NU, MAXIMUM WORDS READ
000374  103405                          BLO     FILL6           ;NO
000376  012767  177777  000004'         MOV     #-1,IEDFI       ;YES,SET MAX FLAG
000404  000167  000116                  JMP     FILLXT          ;GO EXIT
                                ;*
000410                          FILL6:
000410  005000                          CLR     R0              ;SET INDICATOR FOR FIRST BUFFER
000412  004767  000364                  JSR     PC,READ         ;GO READ BLOCK
                                ;*
000416  016702  000454                  MOV     READRS+2,R2
000422  016203  000000                  MOV     CNT(R2),R3      ;SET DATA COUNT
000426  042703  177400                  BIC     #177400,R3      ;CLEAR HIGH BYTE
```

```
000434  001432                          BEQ     FILEOF
000436  016702  000430                  MOV     READ2,R2
000442                          FILL7:
000442  004767  000144                  JSR     PC,STORE            ;SET 4 DATA WORDS
000446  004767  000162                  JSR     PC,MAXV             ;GO FIND MAX VELOCITY
000452  005724                          TST     (R4)+               ;INCREMENT TO NEXT PROCESS WORD
000454  005201                          INC     R1                  ;INCREMENT MAX WORD COUNT
000456  005303                          DEC     R3                  ;FINISHED BLOCK
000460  001370                          BNE     FILL7
                                ;*
000462                          FILLC:
000462  012702  002100'                 MOV     #BUF2,R2
000466  004767  000226                  JSR     PC,TIFR
000472  036227  000002  000040          BIT     BNBT(R2),#EOF       ;END OF FRAME
000500  001010                          BNE     FILEOF              ;YES
000502  020127  001740                  CMP     R1,#MPNTS           ;NO, MAXIMUM POINTS
000506  103657                          BLO     FILL2               ;NO, GO READ NEXT BLOCK
000510  012767  177777  000004'         MOV     #-1,IEOFI           ;YES, GO EXIT
000516  000167  000004                  JMP     FILLXT
000522                          FILEOF:
000522  005267  002372                  INC     FCNTR
000526                          FILLXT:
000526                          FIXET:
000526  005067  000000G                 CLR     BUSY                ;CLEAR WRITE BUSY FLAG
000532  016702  002354                  MOV     REDEX,R2
000536  016762  002342  003106'         MOV     READST,LRSEC(R2)
000544  005362  003100'                 DEC     FTOR(R2)
000550  005301                          DEC     R1
000552  001002                          BNE     FIXT1
000554  000167  177234                  JMP     FILL0
000560                          FIXT1:
000560  010167  000020'                 MOV     R1,NUMPTS
000564                          FIXT:
000564                                  POP
000600  000205                          RTS     R5
000602                          HLT:
000602  012700  000006                  MOV     #6,R0
000606  000167  000000'                 JMP     HALT
```

```
                                      .SBTTL   STORE DATA WORDS
                               ;*
                               ;*     FORMAT AND STORE DATA WORDS
                               ;*
  000612                       STORE:
  000612  012264  000000              MOV      (R2)+,ONE(R4)       ;FIRST DATA WORD   X
  000616  012264  003720              MOV      (R2)+,TWO(R4)       ;SECOND DATA WORD   Y
  000622  012264  007640              MOV      (R2)+,THREE(R4)     ;THIRD DATA WORD  N/I
  000626  012264  013560              MOV      (R2)+,FOUR(R4)      ;FOURTH DATA WORD  V/V
  000632  000207                      RTS      PC                  ;RETURN
```

```
                                        .SBTTL  SET MAXIMUM VELOCITY
                                ;*
                                ;*      FIND MAXIMUM VELOCITY
                                ;*
000634                          MAXV:
000634  026467  013560  002262          CMP     FOUR(R4),MAXW1      ;VELOCITY HIGHER THAN PREVIOUS
000642  101414                          BLOS    MAXV1               ;NO
000644  016767  002254  002254          MOV     MAXW1,MAXW2         ;YES,REPLACE LAST HIGHEST
000652  016767  000000' 000002'         MOV     MAX1,MAX2
000660  016467  013560  002236          MOV     FOUR(R4),MAXW1
000666  010167  000000'                 MOV     R1,MAX1             ;SET CURRENT MAX NUMBER
000672  000411                          BR      MAXV2
                                ;*
000674                          MAXV1:
000674  026467  013560  002224          CMP     FOUR(R4),MAXW2      ;HIGHER THAN PREVIOUS
000702  101405                          BLOS    MAXV2               ;NO
000704  016467  013560  002214          MOV     FOUR(R4),MAXW2      ;YES, REPLACE HIGHEST
000712  010167  000002'                 MOV     R1,MAX2
                                ;*
000716                          MAXV2:
000716  000207                          RTS     PC
                                ;*
                                ;*
000720                          TIFR:
000720                                  PUSH
000734  016267  000014  000010'         MOV     TIMR1(R2),TMEND
000742  016267  000016  000012'         MOV     TIMR2(R2),TMEND+2
000750  016204  000000                  MOV     CNT(R2),R4
000754  072427  177770                  ASH     #-8.,R4
000760  010467  000002'                 MOV     R4,IFRM
000764                                  POP
001000  000207                          RTS     PC
```

```
                                        .SBTTL   READ BLOCK OF DATA
                                ;*
                                ;*      READ NEXT BLOCK OF DATA FROM DISK
                                ;*
001002                          READ:
001002  005767  002106                  TST     FFG
001006  001403                          BEQ     R11
001010  062767  000002  002066          ADD     #2,READST               ;SET READ SECTOR NUMBER
001016                          R11:
001016  016067  001074' 000022          MOV     READRS(R0),READAD       ;SET READ ADDRESS
001024  012767  000001  002062          MOV     #1,FFG
001032  005067  000014                  CLR     CNTR                    ;CLEAR ERROR/FUNCTION
                                ;*
001036  004567  000000G                 JSR     R5,LKTRAN
001042  000000G                         .WORD   FILN
001044  003104'                         .WORD   READST
001046  000000                  READAD: .WORD   0
001050  000004                          .WORD   4                       ;READ
001052  000000                  CNTR:   .WORD   0
                                ;*
001054  005767  177772                  TST     CNTR                    ;ERRORS
001060  001402                          BEQ     READ3                   ;NO, GO EXIT
001062  000167  177440                  JMP     FIXET                   ;ERROR ON LAST READ
                                ;*
001066                          READ3:
001066  000207                          RTS     PC
                                ;*
001070  001120'                 READ1:   .WORD  BUF1+16.
001072  002120'                 READ2:   .WORD  BUF2+16.
001074  001100'                 READRS: .WORD   BUF1                    ;READ BUFFERS
001076  002100'                         .WORD   BUF2
                                ;*
001100                          BUF1:   .BLKW   SIZB
                                        .BLKW   SIZD
                                ;*
002100                          BUF2:   .BLKW   SIZB
                                        .BLKW   SIZD
                                ;*
003100  000000  000000          FTOR:   .WORD   0,0
003104  000000                  READST: .WORD   0
003106  000000  000000          LRSEC:    .WORD 0,0
003112  000000                  REDEX:   .WORD  0
003114  000000                  FFG:    .WORD   0
003116  000000                  FFLG:   .WORD   0                       ;FIRST BLOCK FLAG
003120  000000                  FCNTR:  .WORD   0                       ;NO. FRAMES READ
003122  000000                  FCNTW:  .WORD   0                       ;NO. FRAMES WRITTEN
003124  000000                  MAXW1:  .WORD   0                       ;POINTER TO MAX1 VELOCITY
003126  000000                  MAXW2:  .WORD   0                       ;POINTER TO MAX2 VELOCITY
003130  000022'                 PRODAT: .WORD   LDVDAT+18.              ;PROCESS BUFFER ADDRESS
                                ;*
        000000'                         .CSECT  ABTERM
000000  000000                  ABTRM:  .WORD   0
        000000'                         .CSECT  IFLDV
000000                          IFLD:   .BLKW   1
                                ;*
        000000'                         .CSECT  IHDLI
000000  000000                  MAX1:   .WORD   0
```

```
   000002  000000                     MAX2:    .WORD    0
   000004  000000                     IEOFI:   .WORD    0
           000000'                             .CSECT   LDVDAT
           000000'                    LDVDAT =.
   000000  000000                     IFLY:    .WORD    0
   000002  000000                     IFRM:    .WORD    0
   000004  000000  000000             TMINT:   .WORD    0,0
   000010  000000  000000             TMEND:   .WORD    0,0
   000014  000000                     IDAY:    .WORD    0
   000016  000000                     IPLN:    .WORD    0
   000020  000000                     NUMPTS:  .WORD    0
           000001'                             .END
```

SYMBOL TABLE

```
ABTRM    000000R     002  BB     = 000200           BNBT   = 000002
BUF1     001100RG         BUF2     002100R          BUSY   = ****** G
CNT    = 000000           CNTR     001052R          EOF    = 000040
FCNTR    003120RG         FCNTW    003122RG         FFG      003114R
FFLG     003116R          FILEOF   000522R          FILL     000000RG
FILLA    000360R          FILLB    000322R          FILLC    000462R
FILLE    000060R          FILLE1   000064R          FILLXT   000526R
FILL0    000014R          FILL1    000070R          FILL2    000246R
FILL3    000302R          FILL6    000410R          FILL7    000442R
FILN   = ****** G         FIN    = ****** G         FIXET    000526R
FIXT     000564R          FIXT1    000560R          FNNP   = 000006
FOUR   = 013560           FTOR     003100RG         HDRW   = 000004
HLT      000602R          IDAY     000014R     005  IEOFI    000004R     004
IFLD     000000RG    003  IFLY     000000R     005  IFRM     000002R     005
IPLN     000016R     005  LDVDAT=  000000R     005  LEVW   = 000340
LKTRAN=  ****** G         LNKBLK=  ****** G         LRSEC    003106RG
MAXV     000634R          MAXV1    000674R          MAXV2    000716R
MAXW1    003124R          MAXW2    003126R          MAX1     000000R     004
MAX2     000002R     004  MPNTS  = 001740           NEXT   = ****** G
NUMPTS   000020R     005  ONE    = 000000           PC     =%000007
PRODAT   003130R          PSW    = 177776           READ     001002R
READAD   001046R          READRS   001074R          READST   003104R
READ1    001070R          READ2    001072R          READ3    001066R
REDEX    003112R          REQUST=  ****** G         R0     =%000000
R1     =%000001           R11      001016R          R2     =%000002
R3     =%000003           R4     =%000004           R5     =%000005
R6     =%000006           R7     =%000007           SECT2    000142R
SECT3    000150R          SECT4    000152R          SECT5    000106R
SECT6    000132R          SIZB   = 000010           SIZD   = 000370
SP     =%000006           STORE    000612R          SWR    = 177570
THREE  = 007640           TIFR     000720R          TIMR1  = 000014
TIMR2  = 000016           TIMS1  = 000010           TIMS2  = 000012
TMEND    000010R     005  TMINT    000004R     005  TWO    = 003720
WRITE  = ****** G
. ABS.   000000      000
         003132      001
ABTERM   000002      002
IFLDV    000002      003
IHDLI    000006      004
LDVDAT   000022      005

ERRORS DETECTED:  0
FREE CORE:  11744. WORDS
FILL,L2<FILL.2
```

APPENDIX D

FORTRAN LISTINGS

PRECEDING PAGE BLANK NOT FILMED

```
      SUBROUTINE SCAT
      INTEGER PLTSYM
      DIMENSION LBUF(15)
      DIMENSION IMX(10),ISYM(10)
      DIMENSION BK(11)
      DIMENSION PLTSYM(10),IBUF(6)
      COMMON XCG(2),YCG(2),INDEX(250),NCORPT(2),NOISES(2),PKVEL(2)
     1,ELVANG(2),NVORTX,KFRM,RTIME
      COMMON/BUFFER/IBUF1(20)
      COMMON/TMPS/KDFTT(6),JAUTO
      COMMON/DSPL/NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON/ABTERM/IABTRM
      COMMON/INPT/PTTOL,NOISE,VELTOL,VORTOL,IRADI,IRADI2,
     1 NRADI,CONXY,STIME,FRVTOL
      COMMON /LDVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1   IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON/IHDLI/MAX1,MAX2,IEOFI
      DATA IXOFF,IYOFF,XSTRT,YSTRT,RLX,RLY,IXL,IYL/
     1 140,173,200.,0.,700.,400.,784,442/
       DATA IVCHAR/'V'/
      DATA ISYM/'A','B','C','D','E','F','G','H','I','J'/
      DATA PLTSYM/'0','1','2','3','4','5','6','7','8','9'/
      DATA BK/20.,30.,40.,50.,60.,70.,80.,90.,100.,110.,120./
      DATA IBUF/1,1,0,0,1,0/
      CALL OPEN(1,1)
      XSCAL=IXL/RLX
      YSCAL=IYL/RLY
      IBUF(1)=NS4
      KTWO=2
      CALL DISPIO(14,KTWO,KLG)
C
C     OUTPUT HEADER ON AUXILIARY SCREEN
      NST=2
      IF(NS4.EQ.2)NST=1
      CALL DISPIO(11,NST)
      CALL DISPIO(1,36,IBUF1)
C
C     RE-SELECT SCREEN FOR SCATTER
      CALL DISPIO(11,NS4)
C
 22   CONTINUE
      CALL DBUG
C
C     BRING UP BACKGROUND
      CALL DISPIO(2,13,0)
C
C     ENCODE FRAME FOR SCATTER HEADER
      KFRM=IABS(IFRM)
      CALL DECD(KFRM,2,IBUF1(19))
      IF(IFRM.LT.0)IBUF1(20)=PLTSYM(3)
      IF(IFRM.GE.0)IBUF1(20)=PLTSYM(2)
C
C     BRING UP SCATTER HEADER
      CALL DISPIO(1,40,IBUF1)
C
C     STORE INDICES OF HIGHEST 10 VELOCITIES IN ARRAY IMX
      DO 30 I=1,10
 30   IMX(I)=0
```

ORIGINAL PAGE IS OF POOR QUALITY

```
      DO 31 K=1,10
      IVLM=0
      DO 31 I=1,NUMPTS
      IF(IVEL(I).LE.IVLM)GO TO 31
      IF(K.EQ.1)GO TO 33
      KM1=K-1
      DO 32 J=1,KM1
      IF(I.EQ.IMX(J))GO TO 31
 32   CONTINUE
 33   IMX(K)=I
      IVLM=IVEL(I)
 31   CONTINUE
C
C     ENCODE VELOCITIES
      DO 40 I=1,10,2
      J=11-I
      J=IMX(J)
      IDUM=0
      IF(J.EQ.0)GO TO 300
      CALL GETVEL(IVEL(J),VEL)
      IDUM=VEL
 300  CONTINUE
      K=(31-3*I)/2
      CALL DECD(IDUM,4,LBUF(K))
      J=10-I
      J=IMX(J)
      IDUM=0
      IF(J.EQ.0)GO TO 301
      CALL GETVEL(IVEL(J),VEL)
      IDUM=VEL
 301  CONTINUE
      K=K-1
      CALL DECD(IDUM,3,LBUF(K))
 40   CONTINUE
C
C     OUTPUT VELOCITIES
      CALL DISPIO(2,24,-1)
      CALL DISPIO(1,30,LBUF)
C
C     SELECT CHARACTER TO BE OUTPUT FOR EACH POINT AND OUTPUT
      KTWO=2
      CALL DISPIO(14,KTWO,KSM)
      J=2
      DO 1 I=1,NUMPTS
      CALL GETVEL(IVEL(I),VEL)
      DO 34 L=1,10
      IF(I.NE.IMX(L))GO TO 34
      ICHAR=ISYM(L)
      GO TO 5
 34   CONTINUE
 9    J=J-1
      IF(J)6,6,2
 2    IF(VEL.LE.BK(J))GO TO 9
 3    J=J+1
      IF(J-11)4,4,7
 4    IF(VEL.GT.BK(J))GO TO 3
 8    ICHAR=PLTSYM(J-1)
      GO TO 5
 6    J=2
      GO TO 1
 7    J=11
      GO TO 8
 5    CONTINUE
```

```
      XF=IX(I)*CONXY
      YF=IY(I)*CONXY
      IBUF(5)=1
      IBUF(3)=IXOFF+(XF-XSTRT)*XSCAL
      IBUF(4)=IYOFF+(YF-YSTRT)*YSCAL
      CALL SETAD(IBUF(6),ICHAR)
      CALL DISPIO(4,12,IBUF)
 1    CONTINUE
C
C     CALCULATE CENTROID AND PLOT VORTEX POSITION WITH V
      KTWO=2
      CALL DISPIO(14,KTWO,KSM)
      CALL CENTRD
      KTWO=2
      CALL DISPIO(14,KTWO,KLG)
      IF(NVORTX.EQ.0)GO TO 60
      DO 61 I=1,2
      IF(XCG(I).EQ.0.)GO TO 61
      IBUF(3)=IXOFF+(XCG(I)-XSTRT)*XSCAL
      IBUF(4)=IYOFF+(YCG(I)-YSTRT)*YSCAL
      CALL SETAD(IBUF(6),IVCHAR)
      CALL DISPIO(4,12,IBUF)
 61   CONTINUE
 60   CONTINUE
C
C     HARD COPY IF REQUESTED
      IF(JAUTO.EQ.1)GO TO 65
      CALL DISPIO(10,NS4)
C
C     DELAY
      DO 11 I=1,65
      DO 11 J=1,2000
 11   BDUMB=(ADUMB+2.5)/3.6
 65   CONTINUE
C
C     READ RECORD FROM TAPE
      CALL VREAD
C
C     EXIT IF END OF FILE
      IF(IEOFI.EQ.1)GO TO 20
C
C     EXIT IF !
      IF(IABTRM.EQ.33)GO TO 20
      GO TO 22
 20   RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE DECD (KK,KKK,KP)
      BYTE KP(5)
      BYTE NEG
      DATA NEG/'-'/
      II=0
      K=0
      KNM=KKK
      IF (KK) 10,25,25
   10 KK=-KK
      KP(1)=NEG
      II=1
      KNM=KNM-1
   25 IF(KNM-5) 35,50,200
   35 K=KK/10**KNM
   50 DO 100 I=1,KNM
      KT=K
      K=KK/10**(KNM-I)
  100 KP(I+II)=K-KT*10+48
  200 RETURN
      END
```

```
      SUBROUTINE ONCE
      COMMON /BUFFR/IBFR(12)
      COMMON /DSPL/ NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON/ IADDRS/ IADDXY(26), IADPTS(2),IDATA(66), IADDAT,BLANK,
     1 IABLNK
      COMMON/INPT/  PTTOL,NOISE, VELTOL, VORTOL, IRADI,IRADI2,NRADI,
     1   CONXY,STIME,FRVTOL
      COMMON /OWT/ IOWT(6)
      COMMON /SETUP/ DATE,JDFT(20,6),JPLTP(3),IPLIND,IL1(60),TIM(2),
     1 IFLB(3),ILDV,IAHC,IL2(90),KRBUF(6)
      BYTE IDATA
      INTEGER BLANK
      BYTE XYSYM(52), PLTSYM(3)
      EQUIVALENCE (PLTS1,PLTSYM(1)),(PLTS3,PLTSYM(3))
      EQUIVALENCE (IADPT1,IADPTS(1)),(IADPT2,IADPTS(2))
      EQUIVALENCE (IBFR6,IBFR(6)),(IBFR12,IBFR(12))
      EQUIVALENCE (KRBUF6,KRBUF(6))
      EQUIVALENCE (IOWT6,IOWT(6))
      DATA PLTSYM/ '*', ' ',  'O'/
      DATA XYSYM/'A',' ','B',' ','C',' ','D',' ','E',' ','F',' ',
     1           'G',' ','H',' ','I',' ','J',' ','K',' ','L',' ',
     2           'M',' ','N',' ','O',' ','P',' ','Q',' ','R',' ',
     3           'S',' ','T',' ','U',' ','V',' ','W',' ','X',' ',
     4           'Y',' ','Z',' '/
      IRADI=IRADI/CONXY
      KTWO=2
      CALL DISPIO (14,KTWO,KLG)
      CALL SETAD(IABLNK,BLANK)
      DO 1111 JSYM=1,26
      CALL SETAD(IADDXY(JSYM),XYSYM(JSYM*2-1))
1111  CONTINUE
      CALL SETAD (IADPT1,PLTS1)
      CALL SETAD (IADPT2,PLTS3)
      CALL SETAD (IADDAT,IDATA)
      CALL SETAD (IOWT6,IL1)
      CALL SETAD (KRBUF6,IL2)
      CALL SETAD (IBFR6,DATE)
      CALL SETAD (IBFR12,IFLB)
      RETURN
      END
```

```
      SUBROUTINE VREAD
C
C        THIS SUBROUTINE READS THE VORTEX DATA TAPE FOR POST-PROCESSING
C
      COMMON/ BUFFER/ IBUF1(18)
      COMMON/IDUMMY/NBA,NFL
      COMMON /IFLDV/IFLDV
      COMMON /CFLI/ CFLI
      COMMON /IHDLI/ MAX1,MAX2,IEOFI
      COMMON/INPT/  PTTOL,NOISE, VELTOL, VORTOL, IRADI,IRADI2,NRADI,
     1  CONXY,STIME,FRVTOL
      COMMON /LDVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON/ PLNAME/ IDAT(60)
      COMMON /ROUT/ ROUT,KGBF
      COMMON /SETUP/ DATE,JDFT(20,6),JPLTP(3),IPLIND,IL1(60),TIM(2),
     1 IFLB(3),ILDV,IAHC,IL2(90),KRBUF(6)
      INTEGER CFLI
      DIMENSION BUF1(2), BUF2(2)
C
      DATA BUF1/'TAPE',' ERR'/
      DATA BUF2/'EOF ','READ'/
      DATA DATS/'VRED'/
      ROUT = DATS
C        READ PHYSICAL RECORD FROM TAPE MT1
20    CONTINUE
      CALL GET (1,1,IFLY,4010)
C        WAIT FOR I/O OPERATION TO COMPLETE ON MT1
      CALL WAIT (1,0,N)
C        GET RETURN STATUS FOR END-OF-FILE INDICATOR
      CALL SFUN(1,7,NBA,IEOFI)
      IF (IEOFI) 35,35,30
30    CONTINUE
      CALL DISPIO(3,8,BUF2)
      RETURN
35    CONTINUE
C        CHECK FOR 4008 WORD PHYSICAL RECORD READ
      IF(N-1) 50,50,40
40    CALL DISPIO(3,8,BUF1)
      RETURN
50    CONTINUE
      IF (((IFRM.GT.0)  .AND.  (IFLDV.EQ.1))  .OR.  ((IFRM.LT.0)  .AND.
     1 (IFLDV .EQ.2))) GO TO 20
      DO 60 I=1,NUMPTS
60    IVEL(I)= IABS(IVEL(I))
C     ENCODE FLY-BY NO
      CALL DECD (IFLY,5,IFLB)
C     ENCODE DAY
      CALL DECD(IDAY,3,DATE)
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE DBUG
      COMMON /ABTERM/ IABTRM
      COMMON            XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
     1                   PKVEL(2),ELVANG(2),NVORTX, KFRM,RTIME
      COMMON /ATLP/ IBASE,DIMX
      COMMON /BUFFER/ IBUF1(12),IBUF2(6),IBUF3(2)
      COMMON /BUFFR/ IBFR(12)
      COMMON /CNTRL/ KPROC, IACQTL, MTIRF
      COMMON /DSPL/ NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON/ IADDRS/ IADDXY(26), IADPTS(2),IDATA(66), IADDAT,BLANK,
     1 IABLNK
      COMMON /IHDLI/ MAX1,MAX2,IEOFI
      COMMON /IFLDV/ IFLDV
      COMMON/INPT/  PTTOL,NOISE, VELTOL, VORTOL, IRADI,IRADI2,NRADI,
     1  CONXY,STIME,FRVTOL
      COMMON/INITL/ ITPLT, PORTS(2), STARBS(2),VELMNF
      COMMON/KSTFL/KSTFL,KPLN
      COMMON /LDVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON /OWT/ IOWT(6)
      COMMON /PLNAME/ IDAT(60)
      COMMON /ROUT/ ROUT,KGBF
      COMMON /SEARCH/ ISFBN,ISDAY,ISPN
      COMMON /SETUP/ DATE,JDFT(20,6),JPLTP(3),IPLIND,IL1(60),TIM(2),
     1 IFLB(3),ILDV,IAHC,IL2(90),KRBUF(6)
      COMMON /TMPS/  KDFTT(6), JAUTO
      COMMON /XCORDT/KX1,KX2,KX3,KX4
      BYTE IDATA
      INTEGER BLANK
      DATA M/6/
      CALL SSWTCH (0,K)
      IF (K .NE. 1) RETURN
      CALL DISPIO (7,1)
      KTWO=2
      CALL DISPIO(14,KTWO,KSM)
      CALL DISPIO (13,32)
      WRITE(M,100) IFLY,IFRM,ITMINT    ,ITMEND    ,IDAY,IPLN,NUMPTS
  100 FORMAT(5X,'IFLY=',I6,5X,'IFRM=',I6,5X,'FIRST TIME =',2(I6,1X),4X,
     1 'LAST TIME=',2(I6,1X),/5X,'IDAY=',I6,' IPLN=',I6,' NUMPTS= ',I6,/
     2 //13X,'X                    Y                    I/N            VP/VM'/)
      K1=NUMPTS
      DO 346 I=1,K1
      CALL SUBBIT (INTENS(I),2,8,KK1)
      CALL SUBBIT (INTENS(I),10,7,KK2)
      CALL SUBBIT (IABS(IVEL(I)),2,7,KK3)
      CALL SUBBIT (IABS(IVEL(I)),10,7,KK4)
      WRITE (M,101) I,IX(I),IY(I),KK1,KK2,KK3,KK4
  346 CONTINUE
  101 FORMAT (1X,I6,13X,I6,11X,I6,4X,I6,4X,I6,4X,I6,4X,I6)
  999 CONTINUE
      CALL SSWTCH(1,K)
      IF(K.EQ.1) GO TO 850
      ENDFILE M
850   CONTINUE
      CALL DISPIO (13,64)

      IF((JAUTO.NE.1).AND.(K.NE.1)) CALL DISPIO(10,NS4)
      RETURN
      END
```

```
      BLOCKDATA
       COMMON           XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
     1                    PKVEL(2),ELVANG(2),NVORTX, KFRM,RTIME
      COMMON /ABTERM/ IABTRM
      COMMON /ATLP/ IBASE,DIMX
      COMMON /BUFFER/ IBUF1(12),IBUF2(6),IBUF3(2)
      COMMON /BUFFR/ IBFR(12)
      COMMON /CNTRL/ KPROC, TACOTL, MTIRF
      COMMON /DSPL/ NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON/ IADDRS/ IADDXY(26), IADPTS(2),IDATA(66), IADDAT,BLANK,
     1 IABLNK
      COMMON /IHDLI/ MAX1,MAX2, IEOFI
      COMMON /IFLDV/ IFLDV
      COMMON/INPT/  PTTOL,NOISE, VELTOL, VORTOL, IRADI,IRADI2,NRADI,
     1  CONXY,STIME,FRVTOL
      COMMON/INITL/ ITPLT, PORTS(2), STARBS(2),VELMNF
      COMMON/KSTFL/KSTFL,KPLN
      COMMON /LDVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
       COMMON /OWT/ IOWT(6)
      COMMON /PLNAME/ IDAT(60)
      COMMON /ROUT/ ROUT,KGBF
      COMMON /SETUP/ DATE,JDFT(20,6),JPLTR(3),IPLIND,IL1(60),TIM(2),
     1 IFLB(3),ILDV,IAHC,IL2(90),KRBUF(6)
      COMMON /TMPS/  KDFTT(6), JAUTO
      COMMON /XCORDT/KX1,KX2,KX3,KX4
      COMMON/IREAD/IREAD
      COMMON/MTWFLG/MTWFLG
      BYTE IDATA
      INTEGER BLANK
C        CONXY = 2099./1023.     (640 METERS)
      DATA CONXY/2.0518084/
      DATA ITMINT,ITMEND/0,0,0,150/
      DATA KSM,KLG/035433,034033/
      DATA IREAD/0/
      DATA MTWFLG/0/
      DATA IFLDV /0/
      DATA IDATA/ 66*' '/
      DATA BLANK/'  '/
      DATA ROUT/'BL  '/
      DATA IFLB/2*'00','0 '/
      DATA DATE/'      '/
      DATA IL1/420,627,644,627,-1,420,621,644,621,-1,420,517,700,517,
     1 -1,420,510,700,510,-1,420,473,700,473,-1,420,466,700,466,-1,
     2     420,429,700,429,-1,420,422,700,422,-1,420,319,700,319,-1,
     3     420,312,700,312,-1,420,275,700,275,-1,420,268,700,268,-1/
      DATA PTTOL,NOISE,VELTOL,VORTOL,IRADI,FRVTOL/0.50,2,0.50,0.50,40,
     1 0.50/
      DATA ILDV,IAHC/0,1/
      DATA IPLIND/12/
      DATA JDFT /20*40,20*50,20*50,20*20,20*25,20*50/
      DATA NS1,NS2,NS3,NS4/2,1,3,3/
      DATA LLDT/12/
      DATA KX1,KX2,KX3,KX4/560,700,770,910/
      DATA TACOTL/1.0E+06/
      DATA MTIRF/0/
```

```
      DATA IL2 /910,583,980,583,-1,910,576,980,576,-1,560,473,700,473,
     1 -1,560,466,700,466,-1,560,429,700,429,-1,560,422,700,422,-1,
     2      560,385,700,385,-1,560,378,700,378,-1,770,275,910,275,-1,
     3      770,268,910,268,-1,560,231,700,231,-1,560,224,700,224,-1,
     4      560,187,560,187,-1,560,180,560,180,-1,560,143,560,143,-1,
     5      560,136,560,136,-1,630,143,700,143,-1,630,136,700,136,-1/
      DATA IOWT/1,-1,0,0,120,0/
      DATA IBFR/1,1,308,657,3,0,1,1,406,614,5,0/
      DATA IBASE,DIMX/566,280.0/
      DATA KRBUF/1,-1,0,0,180,0/
      DATA IDAT/'B-','70','7 ','L-','10','11','2 ','E ','JT','B-','72',
     1       '7 ','C-','88','0 ','4 ','E ','JT','B-','73','7 ','VC','-1',
     2       '0 ','2 ','E ','PR','B-','74','7 ','BA','C1','11','1 ','E ',
     2       'PR','DC','-8','  ','A-','30','0 ','DT','HE','R ','DC','-9',
     3       '  ','IL','YU','SN','  ','  ','  ','DC','-1','0 ','YA','C-',
     4       '40'/
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE STRT (N)
       COMMON            XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
     1                    PKVEL(2),ELVANG(2),NVORTX, KFRM,RTIME
      COMMON /ABTERM/ IABTRM
      COMMON /BUFFER/ IBUF1(20)
      COMMON/CFLI/CFLI
      COMMON /CNTRL/ KPROC, TACQTL, MTIRF
      COMMON /DSPL/ NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON /IHDLI/ MAX1,MAX2,IEOFI
      COMMON/INITL/ ITPLT, PORTS(2), STARBS(2),VELMNF
      COMMON/INPT/, PTTOL,NOISE, VELTOL, VORTOL, IRADI,IRADI2,NRADI,
     1  CONXY,STIME,FRVTOL
      COMMON/IREAD/IREAD
      COMMON/KSTFL/KSTFL,KPLN
      COMMON/KTFLG/KTFLG
      COMMON /LDVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON/MTWFLG/MTWFLG
      COMMON /ROUT/ ROUT,KGBF
      COMMON /SETUP/ DATE,JDFT(20,6),JPLTP(3),IPLIND,IL1(60),TIM(2),
     1 IFLB(3),ILDV,IAHC,IL2(90),KRBUF(6)
      COMMON /STAT/ TOT(8), KVAN1, KVAN2
      COMMON /TMPS/  KDFTT(6), JAUTO
      BYTE CHAR(6)
      BYTE IDATE(4)
      INTEGER CFLI
      DIMENSION ERMSG(3)
      DIMENSION IBUF(6)
      DIMENSION ISTIM(2)
      DIMENSION N(6)
      DIMENSION NNN(2)
      DIMENSION TEXT(16), TEXT1(16)
      EQUIVALENCE (CHAR(1),IFLB(1))
      EQUIVALENCE (IDATE(1),DATE)
      DATA DATS/'STRT'/
      DATA ERMSG /'OPT ', 'TAPE',' ERR'/
      DATA IBUF/1,1, 0,0,64,0/
      DATA KLGFM /0/
      DATA TEXT/'FR D','ATA ',' COR',' PT ',' NOI','SE  ','ANGL','E  P',
     1 'K  V','EL T','IME ','  PO','RT P','OS S','TARB',' POS'/
      DATA TEXT1/'   P','TS  ','  P ',' S  ','  P ','S   ','MN M','X   '
     1 ,'P   ','S   ','    ','   X','   Y','    ',' X  ',' Y  '/
C
C         STRT SUBROUTINE PROCESSING
C
C               VORTEX - COMMON VARIABLES DEFINITION
C
C         CFLI   - CONTINUOUS MODE FLAG (0-OPERATOR CONTROLLED), 1-CONTINUOUS)
C         CHAR   - BYTE ARRAY: EQUIVALENCE (CHAR,IFLB)
C         DATS   - MNUEMONIC OF SUBROUTINE CODE
C         IABTRM - FLAG TO INDICATE KEYBOARD REQUEST
C                  * (ABORT)        [42\
C                  ] (END FLY-BY+ [33\
C         IEOFI  - FLAG FOR END OF FLY-BY (0-NORMAL FRAME,1-END-OF-FLY-BY;
C         IFLY   - FLY-BY NUMBER
C         IRADI  - CORRELATION CIRCLE RADIUS
C         IRADI2 - SQUARE OR CORRELATION CIRCLE RADIUS
```

```
C          ISTIM  - ARRAY  START FLY-BY TIME
C          ITPLT  -
C          KFRM   - POSITIVE FRAME NUMBER
C          KLGFM  - FLAG TO INDICATE NUMBER OF DATA PTS EXCEED 992.
C          KPROC  - FLAG FOR TYPE OF PROCESSING (1-REAL TIME, 2-POST ANALYSIS
C          KSTFL  - FLAG FOR START FLY-BY INTERRUPT (1-INTERRUPT)
C          LDIT   - Y DIT POSITION OF CURRENT LINE OF TABULAR OUTPUT
C          LLDIT  - INCREMENT OF Y-DIT POSITION FOR TABULAR LINES
C          LFRAME - MAXIMUM FOR CURRENT PAGE OF TABULAR DATA
C          MITRF  - FLAG FOR MAG TAPE WRITE REQUEST (0-WRITE, 1-INHIBIT)
C          N      - ARRAY: TABLET OPTION SELECTION
C          N(1)   - DISPLAY NUMBER
C          N(2)   - OPTION NUMBER
C          NRADI  - NEGATIVE OF CORRELATION CIRCLE RADIUS
C          NS1    - SCREEN ASSIGNMENT FOR X-Y PLOTS
C          NS2    - SCREEN ASSIGNMENT FOR TIME BASED PLOTS
C          NS3    - SCREEN ASSIGNMENT FOR TABULAR DATA
C          NS4    - SCREEN ASSIGNMENT FOR SCATTER PLOTS
C          PORTS  - ARRAY:
C          ROUT   - MNEUMONIC OF CURRENT SUBROUTINE
C          RTIME  - TIME SINCE START OF FLY-BY
C          STARBS - ARRAY
C          STIME  - START FLY-BY TIME
C          TACQTL - TIME LIMIT FOR FLY-BY (SECONDS)
C          VELMNF -
C
      IABTRM=0
C     MTWFLG SET=0 BLOCK DATA   SET=1 IN STRT WHEN TAPE UNIT 0 IS OPENED
      IF(MTWFLG) 10,3000,10
C     MTIRF  SET=0 IN BLOCK DATA SET=I RTV FOR POST ANAL, SET=1 IN
C            DSEL IF TAPE REC INHIBITTED IN REAL TIME
3000  IF(MTIRF) 10,3002,10
3002  CONTINUE
C        OPEN MAG TAPE (MTO) FOR OUTPUT
      CALL OPEN (2,0)
      CALL SFUN(0,11,NBA,NFL)
      NBA=1
      CALL SFUN(0,5,NBA,NFL)
      MTWFLG=1
10    CONTINUE
      IREAD=0
C
C        SET CONTINUOUS OR OPERATOR CONTROLLED START FLY-BY
      IF (CFLI) 22,11,11
   11 CONTINUE
      IF(N(1).EQ.3) CFLI=1
      IF(N(1).NE.6) GO TO 2
      IF(N(2).EQ.7) CFLI=0
      IF(N(2).EQ.8) CFLI=1
2     CONTINUE
      IF(N(1).EQ.10) CFLI=0
      IF(N(1).EQ.16) CFLI=0
C
C        IF CONTINUOUS MODE ENABLE START FLY-BY INTERRUPTS
      IF (CFLI .EQ. 1) CALL INIT1
   22 CONTINUE
      ROUT = DATS
C        TERMINATE REFRESH MODE
```

```
      CALL DISPIO (13,1)
C     INITIALIZE PORTS AND STARBS FOR 1ST FRAME
      PORTS(1)=0
      PORTS(2)=800
      STARBS(1)=800
      STARBS(2)=0
      VELMNF=0.0
      ITPLT=0
      IRADI2=IRADI*IRADI
      NRADI=-IRADI
      DO 4317 I=1,8
 4317 TOT(I)=0.0
      KVAN1=0
      KVAN2=0
C
C         BRING UP DISPLAY BACKGROUNDS
      IF (NS1 .EQ. 3) GO TO 75
C
C         BRING UP X-Y PLOT BACKGROUND
C         SELECT SCREEN NS1
      CALL DISPIO (11,NS1)
      CALL DISPIO (2,7,8)
   75 CONTINUE
      IF (NS2 .EQ. 3) GO TO 80
C
C         BRING UP TIME PLOT BACKGROUND
C         SELECT SCREEN NS2
      CALL DISPIO (11,NS2)
      CALL DISPIO (2,11,12)
   80 CONTINUE
      KFRM=0
81    CONTINUE
C
C         INITIALIZE COUNT FOR MAXIMUM TABULAR SIZE
      LFRAME=KFRM+55
      IF (NS3 .EQ. 3) GO TO 85
C
C         BRING UP TABULAR HEADING
C         SELECT SCREEN NS3
      CALL DISPIO (11,NS3)
      CALL DISPIO (2,19,0)
      LDT = 712
C         PRINT TABULAR HEADING
      IBUF(1) = NS3
      IBUF(4) = LDT
      CALL SETAD (IBUF(6),TEXT)
      IBUF(3)=512
      CALL DISPIO(4,12,IBUF)
C
      IBUF(5)=64
      LDT = LDT - LLDT
      IBUF(4) = LDT
      CALL SETAD (IBUF(6),TEXT1)
      CALL DISPIO(4,12,IBUF)
      LDT = LDT - LLDT
   85 CONTINUE
      TABTRM=0
      IF (CELI .EQ. 1) GO TO 850
```

```
C
C          IF NOT CONTINUOUS MODE CREATE HEADER FILL-IN
      N(2)=10
      CALL DFLT (N)
  850 CONTINUE
C
C          PROCESS DATA
C
C          CHECK FOR ADDITIONAL PAGES OF TABULAR DATA
      IF (LFRAME .GT. 60) GO TO 894
C          CHECK FOR OPERATOR CONTROLLED MODE
      IF (CFLI) 891,891,857
  857 CONTINUE
C
C          IF CONTINUOUS MODE LOOK FOR START FLYBY INTERRUPT
C
C          STOP -- WAIT FOR START FLY-BY INTERRUPT
C          IF IABTRM=42 ABORT FLY-BY
 89   IF (IABTRM .EQ. 42) GO TO 91
      IF (KSTFL .NE. 1) GO TO 89
C
C          DECODE BCD-CODED PLANE TYPE
      I1=KPLN/16
      IF (KPLN-16*I1 .GT. 9) KPLN=16*I1+9
      N(2)=KPLN-6*I1+1
      N(4)=0
      CALL PTYP (N)
      N(2)=10
      CALL DFLT (N)
C
C          RESET START FLY-BY INTERRUPT
  891 KSTFL=0
C          BRING UP SCATTER PLOTS
      IF (NS4 .EQ. 3) GO TO 87
      CALL SCAT
      GO TO 95
   87 CONTINUE
C
C          SELECT LARGE CHARACTER SIZE
 894  KTWO=2
      CALL DISPIO(14,KTWO,KLG)
C
C          BRING UP HEADER ON SCREENS 1 AND 2
C          SELECT SCREEN J
      CALL DISPIO (11,3)
C          PRINT HEADER-FILL IN
 893  CALL DISPIO(1,36,IBUF1)
C          SELECT SMALL CHARACTER SIZE
      KTWO=2
      CALL DISPIO(14,KTWO,KSM)
C
C          IF OTHER THAN FIRST PAGE OF TABULAR DATA SKIP INITIALIZATION
      IF (LFRAME .GT. 55) GO TO 90
      KTFLG=0
      IF (KPROC .EQ. 2) GO TO 90
```

```
C
C          IF REAL-TIME MODE ENABLE DATA & END-OF-FRAME INTERRUPTS
      CALL INIT
C
C          CALCULATE FLY-BY NUMBER FOR TAPE
      IFLY=0
      DO 892 I=1,5
      IFLY=IFLY*10+CHAR(I)-48
 892  CONTINUE
C          DECODE DAY NUMBER FOR TAPE
      IDAY=0
      DO 600 I=1,3
      IDAY=IDAY*10+IDATE(I)-48
  600 CONTINUE
      IFLN=IFLIND
C
C          FRAME ENTRY POINT
C
   90 CONTINUE
C
C          CHECK FOR STOP FLY-BY KEYBOARD REQUEST
      IF (IABTRM.EQ. 33) GO TO 95
C
C          CHECK FOR ABORT KEYBOARD REQUEST
      IF (IABTRM.NE. 42) GO TO 92
C
C          ABORT CODE
C          SELECT LARGE CHARACTERS
C          RESET KEYBOARD REQUEST CODE
   91 IABTRM=0
      IF (KPROC.NE. 1) GO TO 911
C          DISABLE START FLY-BY INTERRUPTS
      CALL DISAB1
C          DISABLE DATA & END-OF-FRAME INTERRUPTS
      CALL DISABL
C          SELECT OPERATOR CONTROLLED MODE
  911 CONTINUE
      CFLI=0
      IF (MTWFLG.EQ. 0) GO TO 191
      IF (KPROC.EQ. 2) GO TO 191
      CALL SFUN (0,2,NBA,NFL)
      CALL SFUN (0,2,NBA,NFL)
      NBA=1
      CALL SFUN (0,5,NBA,NFL)
  191 CONTINUE
      CALL DISPIO (11,1)
C          BRING UP INITIAL DISPLAY
      CALL DISPIO (2,1,0)
      GO TO 950
   92 CONTINUE
C
C          IF REAL-TIME MODE FILL PROCESSING BUFFERS WITH FRAME OF DAT
      IF (KPROC.EQ. 1) CALL FILL
C
C          CHECK FOR STOP FLY-BY KEYBOARD REQUEST
      IF (IABTRM.EQ. 33) GO TO 95
C
C          CHECK FOR ABORT KEYBOARD REQUEST
      IF (IABTRM.EQ.42) GO TO 91
```

```
C
C          IF TAPE WRITE OPTION SELECTED INITIATE TAPE I/O
      IF(MTIRF.EQ.0) CALL PUT (0,1,IFLY,4009)
C
C          CHECK FOR END-OF-FLY-BY
      IF (IEOFI)935,93,95
C
C          CHECK FOR FRAME EXCEEDING  992 DATA PTS
   93 IF (KLGFM .NE. 1) GO TO 931
      KLGFM=0
      GO TO 94
  931 CONTINUE
      KFRM=IABS(IFRM)
C
C          CALL CENTROID TO COMPUTE AND OUTPUT VORTEX CENTROIDS
C
      CALL CENTRD
      GO TO 94
C
C          SET FLAG TO INDICATE EXCESSIVE NUMBER OF DATA POINTS
  935 KLGFM=1
   94 CONTINUE
      IF (KPROC .EQ. 1) GO TO 601
      CALL VREAD
      IF (IEOFI .EQ. 1) GO TO 95
  601 CONTINUE
C
C          WAIT FOR PREVIOUS I/O OPERATION ON MTO TO COMPLETE
      K=0
      IF (MTIRF .EQ. 0) CALL WAIT (0,0,K)
C
C          CHECK FOR TAPE ERROR ON TAPE WRITE
      IF (K .NE. 0) CALL DISPIO (3,12,ERMSG)
C
C          CHECK FOR MAXIMUM PAGE COUNT ON TABULAR DATA
      IF (KFRM .LT. LFRAME) GO TO 1094
      IF (JAUTO .NE. 1) CALL DISPIO (10,3)
      GO TO 81
 1094 CONTINUE
C
C          CHECK FOR FLY-BY EXCEEDING TIME LIMIT
      IF (RTIME .LT. TACQTL) GO TO 90
C
C          RESET END-OF-FLY-BY FLAG
   95 IEOFI=0
C
C          IF TAPE WRITE OPTION SELECTED WRITE EOF ON TAPE
      IF (MTIRF .EQ. 0) CALL SFUN (0,2,NBA,NFL)
C
C          IF REAL-TIME MODE DISABLE DATA & END-OF-FRAME INTERRUPTS
      IF (KPROC .EQ. 1) CALL DISABL
C
C          INCREMENT FLY-BY ID BY 1 (ASCII)
      K=5
   96 CONTINUE
      IF (CHAR(K) .LT. 57) GO TO 97
      CHAR(K)=48
```

```
      K=K-1
      GO TO 96
   97 CHAR(K)=CHAR(K)+1
C
C        RESET KEYBOARD REQUEST CODE
      IABTRM=0
C
C        DELAY
C        HARD COPY 4014 IF REQUESTED
      IF (JAUTO .NE. 1) CALL DISPIO (10,3)
C
C        REAL-TIME MODE
      KT=2
      NNN(2)=1
C
C        CHECK FOR POST-ANALYSIS MODE
      IF (KPROC .NE. 2) GO TO 98
      KT=9
      NNN(2)=3
   98 CONTINUE
C
C        CHECK FOR CONTINUOS OPERATION MODE
      IF (CFLI) 22,99,22
   99 CONTINUE
C
C        BRING UP REAL-TIME OPTION SELECTION DISPLAY
C        BRING UP POST-ANALYSIS OPTION SELECTION DISPLAY
      CALL DISPIO (2,KT,0)
      CALL RTV (NNN)
  950 RETURN
      END
```

```
      SUBROUTINE TERM (N)
      COMMON /CNTRL/ KPROC,TACQTL,MTIRF
      COMMON /MTWFLG/ MTWFLG
      IF(KPROC.EQ.2)GO TO 50
      IF (MTWFLG .NE. 1)GO TO 100
      CALL SFUN(0,2,NBA,NFL)
      CALL SFUN(0,2,NBA,NFL)
      CALL SFUN(0,3,NRECD,NFL)
      CALL WAIT(0,0,NW)
      CALL CLOSE(2,0)
  100 CONTINUE
      CALL DISPIO (13,16)
      RETURN
50    CALL SFUN(1,3,NRECD,NFL)
      CALL WAIT(1,0,NW)
      GO TO 100
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE DSEL (N)
      COMMON /LDVDAT/IFLY, IFRM, ITMINT(2), ITMEND(2), IDAY, IPLN, NUMPTS,
     1 IX(1000), IY(1000), INTENS(1000), IVEL(1000)
      COMMON /ABTERM/ IABTRM
      COMMON /ATLP/ IBASE, DIMX
      COMMON /CNTRL/ KPROC, TACQTL, MTIRF
      COMMON /DSPL/ NS1, NS2, NS3, NS4, LDT, KSM, KLG, LLDT
      COMMON /IFLDV/ IFLDV
      COMMON /SETUP/ DATE, JDFT(20,6), JPLTP(3), IPLIND, IL1(60), TIM(2),
     1 IFLB(3), ILDV, IAHC, IL2(90), KRBUF(6)
      COMMON /XCORDT/KX1, KX2, KX3, KX4
      COMMON /TMPS/  KDFTT(6),  JAUTO
      COMMON /CFLI/ CFLI
      INTEGER CFLI
      BYTE CHAR(6), KDABT(4)
      DIMENSION N(8)
      DIMENSION KDATE(2)
      DIMENSION NS(4)
      DIMENSION NNN(4)
      EQUIVALENCE (IFLB(1), CHAR(1)), (KDATE(1), KDABT(1))
      EQUIVALENCE (DATE, KDATE)
      EQUIVALENCE (NS1, NS(1))
      EQUIVALENCE (IL01, IL2(1)), (IL3, IL2(3)), (IL6, IL2(6)), (IL8, IL2(8))
      EQUIVALENCE (IL81, IL2(81)), (IL83 , IL2(83)), (IL86, IL2(86)),
     1 (IL88, IL2(88))
      IF (IABTRM .EQ. 42) GO TO 975
      N2=N(2)
      N5=N(5)
      NT=(N2+1)/2
      ASSIGN 950 TO KST
      IF (N(4) .NE. 1) GO TO 35
      IF (N2 .NE. 1) GO TO 36
      KDATE(1)=N5
      KDATE(2)=N(6)
      IF (KPROC .EQ. 1) GO TO 950
      ISDAY=0
      DO 33 I=1,3
   33 ISDAY=ISDAY*10+KDABT(I)-48
 2000 IF(ISDAY-IDAY)  2500,34,3000
C     NEED TO BACKSPACE FILE
 2500 CONTINUE
      NBA=2
      CALL SFUN(1,13,NBA,NFL)
      IF(NFL) 975,2600,975
 2600 CONTINUE
      CALL VREAD
      GO TO 2000
 3000 CONTINUE
C     FOREWARD SPACE A FILE
      NBA=1
      CALL SFUN(1,10,NBA,NFL)
      IF(NFL) 975,2600,975
  34  CONTINUE
      IPLIND=IPLN
      CFLI=1
```

```
      NNN(2)=IPLIND
      NNN(4)=0
      CALL PTYP (NNN)
      CFLI=0
      GO TO 950
36    CONTINUE
      IFLB(1)=N5
      IFLB(2)=N(6)
      IFLB(3)=N(7)
      IF (KPROC .EQ. 1) GO TO 950
      ISFBN=0
      DO 1501 I=1,5
 1501 ISFBN=ISFBN*10+CHAR(I)-48
4000  IF(ISFBN- IFLY) 4500,6000,5000
4500  CONTINUE
C     NEED TO BACKSPACE FILE
      NBA=2
      CALL SFUN(1,13,NBA,NFL)
      IF(NFL) 975,4600,975
4600  CONTINUE
      CALL VREAD
      GO TO 4000
5000  CONTINUE
C     NEED FOREWARD SPACE A FILE
      NBA=1
      CALL SFUN(1,10,NBA,NFL)
      IF(NFL) 975,4600,975
6000  CONTINUE
      N2=1
      GO TO 34
   35 CONTINUE
C         IBASE IS DIT POSITION OF BAR ORIGIN
C         DIMX IS DELTA DIT RANGE FOR ENTIRE BAR
      GO TO (40,45,50,100,150,200,250,275,300,300,300,300,300,300,300,
     1 300,900,925),N2
   40 CONTINUE
      IF (KPROC .NE. 2) GO TO 44
      CALL SFUN (1,3,NBA,NBA)
      CALL SFUN (1,4,1,NBA)
      CALL VREAD
      GO TO 6000
   44 CONTINUE
      IF (N5 .LT. IBASE) N5=IBASE
      TACQTL = ((N5-IBASE)*300.0)/DIMX
      IL01=IBASE
      IL3=IBASE+(TACQTL*DIMX/300.)
      IL6=IBASE
      IL8=IL3
      GO TO 950
   45 CONTINUE
      IF (KPROC .NE. 2) GO TO 47
      CFLI=-1
      GO TO 950
   47 CONTINUE
      TACQTL=1.0E+6
```

```
      IL01=910
      IL3=980
      IL6=910
      IL8=980
      GO TO 950
50    CONTINUE
      IF (IFLDV .EQ. 1) IFLDV=0
      GO TO 435
100   CONTINUE
      IFLDV=1
      GO TO 460
150   CONTINUE
      IF (IFLDV .EQ. 2) IFLDV=0
      GO TO 435
200   CONTINUE
      IFLDV=2
      GO TO 460
250   CONTINUE
      MTIRF=0
      GO TO 435
275   CONTINUE
      MTIRF=1
      GO TO 460
300   CONTINUE
      KI=1
      IF ((N2.EQ.10) .OR. (N2.EQ.12) .OR. (N2.EQ.14) .OR. (N2.EQ.16))
     1 KI=2
      ASSIGN 300 TO KST
      DO 400 I=1,4
      IF (NS(I) .NE. KI) GO TO 400
      NS(I)=3
      K1=KX1
      K2=KX1
      KT=10*I+31
      GO TO 700
400   CONTINUE
      ASSIGN 950 TO KST
      IF (N2-2*NT) 425,450,450
425   NS(NT-4)=1
435   K1=KX1
      K2=KX2
      GO TO 475
450   NS(NT-4)=2
460   K1=KX3
      K2=KX4
475   KT=NT*10-9
700   IL2(KT)=K1
      IL2(KT+2)=K2
      IL2(KT+5)=K1
      IL2(KT+7)=K2
      GO TO KST,(300,950)
900   CONTINUE
      JAUTO=2
      IL81=630
      IL83=700
      IL86=630
      IL88=700
      GO TO 950
```

```
925   CONTINUE
      JAUTO=1
      IL81=840
      IL83=910
      IL86=840
      IL88=910
 950  CONTINUE
      CALL DISPIO (9,12,KRBUF)
      RETURN
 975  IABTRM=0
      CALL DISPIO (11,1)
      CALL DISPIO (2,1,0)
      RETURN
      END
```

```
      SUBROUTINE CENTRD
C     THIS SUBROUTINE PERFORMS A SEARCH TO LOCATE THE POINTS WHICH
C     DEFINE A VORTEX AND CALCULATES THE CENTROID OF THESE POINTS
C     CONXY = SCALE FACTOR TO CONVERT IX AND IY TO FEET
C     ELVANG(1) ELEVATION ANGLE OF FIRST DATA POINT IN THIS FRAME
C     ELVANG(2) ELEVATION ANGLE OF FIRST DATA POINT IN THIS FRAME
C     INTENS= INTENSITY OF DATA POINT
C     IRADI2= SQUARE OF RADIUS FOR CORRELATION CIRCLE
C     IVEL  = VELOCITY OF DATA POINT
C     IX    = X COORDINATE OF DATA POINT
C     IY    = Y COORDINATE OF DATA POINT
C     KOUNT = NUMBER OF POINTS WHICH LIE IN CORRELATION CIRCLE
C     LSEARC= MAXIMUM NUMBER OF SEARCHES ALLOWED FOR VORTEX LOCATION
C     MAX1  = INDEX OF MAXIMUM VELOCITY IN A FRAME
C     MAX2  = INDEX OF SECOND HIGHEST VELOCITY IN A FRAME
C     MINCNT= NUMBER OF POINTS IN CORRELATION CIRCLE POSSING A PEAK VELOCITY
C             EQUAL TO OR GREATER THAN MINVEL
C     MINVEL= ITOL2*PEAK VELOCITY
C     NOISES= NOISE SPIKES THAT WERE DISCARDED IN PROCESSING DATA FOR A
C             SPECIFIC VORTEX - ARRAY OF 2
C     NSEARC= NUMBER OF SEARCHES THAT HAVE BEEN MADE IN AN ATTEMPT TO
C             LOCATE A VORTEX
C     NSPIKE= NOISE SPIKES THAT ARE FOUND IN A FRAME
C     NUMPTS= NUMBER OF DATA POINTS IN A FRAME
C     NVORTX= NUMBER OF VORTICES THAT HAVE BEEN LOCATED IE. NVORTX + 1
C             IS THE VORTEX NO. THAT IS CURRENTLY BEING PROCESSED
C     PTTOL = PER CENT OF POINTS IN A CORRELATION AREA THAT MUST HAVE A
C             VELOCITY GT VELTOL*PEAK VELOCITY
C     SUM1= SUM OF PRODUCTS OF INTENSITY  AND VELOCITY  OF ALL POINTS
C           IN THE CORRELATION CIRCLE
C     SUM2= SUM OF THE PRODUCTS OF INTENSITY, VELOCITY, AND X COORDINATE
C           OF ALL POINTS IN THE CORRELATION CIRCLE
C     SUM3= SUM OF THE PRODUCTS OF INTENSITY,VELOCITY, AND Y COORDINATE
C           OF ALL POINTS IN THE CORRELATION CIRCLE
C     VELMN2= MINIMUM ACCEPTABLE PEAK VELOCITY
C     VELTOL= VELOCITY TOLERANCE--PER CENT OF PEAK VELOCITY THAT SOME
C             PER CENT OF POINTS IN A CORRELATION AREA MUST POSSESS
C     VORTOL= PER CENT OF PEAK VEL FROM 1ST VORTEX THAT PEAK VEL FROM
C             2ND VORTEX MUST MEET
C     XCG   = X CENTROID  OF VORTEX
C     YCG   = Y CENTROID  OF VORTEX
       COMMON            XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
     1                   PKVEL(2), ELANG1,ELANG2,NVORTX, KFRM,RTIME
      COMMON /CNTRL/ KPROC, TACQTL, MTIRF
      COMMON /DSPL/ NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON/IDUMMY/NBA,NFL
      COMMON /IHOLI/ MAX1,MAX2,IEOFI
      COMMON/INITL/ ITPLT, PORTS(2), STARBS(2),VELMNF
      COMMON/INPT/  PTTOL,NOISE, VELTOL, VORTOL, IRADI,IRADI2,NRADI,
     1  CONXY,STIME,FRVTOL
      COMMON/ KTFLG/KTFLG
      COMMON /LDVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON /ROUT/ ROUT,KGBF
      COMMON /SETUP/ DATE,JDFT(20,6),JPLTP(3),IPLIND,IL1(60),TIM(2),
     1 IFLB(3),ILDV,IAHC,IL2(90),KRBUF(6)
```

```
      COMMON /ABTERM/ IABTRM
      COMMON /STAT/ TOTX1,TOTX2,TOTY1,TOTY2,SSQX1,SSQX2,SSQY1,SSQY2,
     1 KVAN1, KVAN2
      INTEGER KBUFOT(6)
      INTEGER MSTAT(16)
      REAL STTT(8)
      EQUIVALENCE (XCG1,XCG(1)),(YCG1,YCG(1))
      EQUIVALENCE (MSTAT,STTT)
      EQUIVALENCE (MSTAT4,MSTAT(4)),(MSTAT7,MSTAT(7)),(MSTA13,MSTAT(13))
      EQUIVALENCE (MSTA15,MSTAT(15))
      EQUIVALENCE (KBUFT6,KBUFOT(6))
      EQUIVALENCE (WTEST,JPLTP)
      DATA SNAME/ 'CENT'/
      DATA KBUFOT /4*1,32,0/
      DATA STTT /' AVG','=(   ','    ,','       ',' ) S','TD=(','    ,',
     1 '    )'/
      DATA WHEEL /'WHEE'/
      DATA LNOISE/5/
      ROUT= SNAME
      TMINT= 32768.0*ITMINT(1) + ITMINT(2)
      IF(KTFLG) 428,428,429
428   CONTINUE
      STIME=TMINT
      KTFLG=1
429   CONTINUE
      NSPIKE=0
C     INITIALIZE NOISE(1) FOR COMPUTATION OF NOISE FOR EA. VORTEX
      NOISES(1)=0
      NVORTX=0
      MINPTS=2
C     CALCULATE MAXIMUM AND MINIMUM ELATION ANGLES
      X=IX(1)
      Y=IY(1)
      ELANG1    = ATAN2(Y,X)*57.2958
      X=IX(NUMPTS)
      Y=IY(NUMPTS)
      ELANG2    = ATAN2(Y,X)*57.2958
      IF(NUMPTS-1)5000,5000,5010
5000  CONTINUE
      NCORPT(1)=NUMPTS
      CALL GETVEL(IVEL(1),VELMAX)
      GO TO 5020
5010  CONTINUE
      GO TO (1,2000),KPROC
1     CALL GETVEL(IVEL(MAX1),VELMAX)
2     CONTINUE
      IF(VELMAX-VELMNF) 9,10,10
9     CONTINUE
      NCORPT(1)= 0
5020  CONTINUE
      NCORPT(2)= 0
      PKVEL(1) = VELMAX
      GO TO 2500
10    CONTINUE
C     INITIALIZE SUM TERMS AND COUNTERS
      SUM1=0.0
      SUM2=0.0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUM3=0 0
      MINCNT=0
      KOUNT=0
C     COMPUTE MINIMUM VELOCITY FOR CORRELATION CIRCLE
      VELMIN = VELTOL*VELMAX
C     DETERMINE WHICH POINTS LIE IN CORRELATION CIRCLE AND WHICH OF
C     THESE POSSESS THE MINIMUM VELOCITY, COMPUTE SUM TERMS
          NFL=IX(MAX1)
          IYMAX=IY(MAX1)
          DO 100 L=1,NUMPTS
          NBA=IX(L)
          IDX=NBA-NFL
          IF(IDX - IRADI) 30,30,100
30        CONTINUE
          IF(IDX + IRADI)100,35,35
35        CONTINUE
          IYY= IY(L)
          IDY=IYY-IYMAX
          IF(IDY - IRADI)40,40,100
40        CONTINUE
          IF(IDY + IRADI)100,43,43
43        CONTINUE
          IR= IDX*IDX + IDY*IDY
          IF(IR - IRADI2)45,45,100
45        CONTINUE
          CALL GETVEL(IVEL(L) ,VELCTY)
          IF(VELCTY)100,100,48
48        CONTINUE
          IF(VELCTY-VELMIN)50,49,49
49        CONTINUE
          MINCNT = MINCNT + 1
50        CONTINUE
          KOUNT = KOUNT + 1
          CALL SUBBIT(INTENS(L),2,8,IDUM)
          DUM1= IDUM*VELCTY
          SUM1  = SUM1 + DUM1
          SUM2= SUM2 + DUM1*NBA
          SUM3= SUM3 + DUM1*IYY
          INDEX(KOUNT) = L
          IF(KOUNT-250)100,110,110
100       CONTINUE
      IF(KOUNT-MINPTS)1000,110,110
110   CONTINUE
C     DID ENOUGH POINTS HAVE THE MINUMUM VELOCITY, IF NOT TRY ANOTHER
C     CORELATION CIRCLE
      CNTMIN=MINCNT
      IF(CNTMIN-PTTOL*KOUNT)1000,130,130
130   CONTINUE
C     CALCULATE CENTROID  OF ALL POINTS IN CORRELATION CIRCLE
      NVORTX=NVORTX + 1
      DUM1=1 0/SUM1
      DUM2=DUM1*SUM2
      LXCG=DUM2
      XCG(NVORTX)=CONXY*DUM2
      DUM2=DUM1*SUM3
      LYCG=DUM2
```

```
      YCG(NVORTX)=CONXY*DUM2
      NOISES(NVORTX)= NSPIKE - NOISES(1)
      NCORPT(NVORTX)= KOUNT
      VELMNF=FRVTOL*VELMAX
      PKVEL(NVORTX)= VELMAX
      IF(NVORTX-2)425,2500,2500
425   CONTINUE
C     COMPUTE TIME
      TMEND= 32768.0*ITMEND(1) + ITMEND(2)
      ANGLE=ATAN2(YCG(1),XCG(1))*57.2958
C     COMPUTE RTIME FOR START OF THIS SCAN
      RTIME=  .01666666*(TMINT - STIME)
      DTIME =  .01666666*(TMEND - TMINT)
C     CHECK EVEVATION ANGLES FOR DIRECTION OF SCAN
      IF(ELANG2-ELANG1) 430, 460, 440
430   CONTINUE
C     SCANNING DOWN SWAP ANGLES
      NBA=ELANG1
      ELANG1=ELANG2
      ELANG2=NBA
      IF(ANGLE-ELANG2) 435, 435, 460
435   CONTINUE
      IF(ANGLE - ELANG1)500,500,438
438   CONTINUE
      DTIME= DTIME*(ELANG2 - ANGLE)/(ELANG2 - ELANG1)
      GO TO 500
440   CONTINUE
C     SCANNING UP
      IF(ANGLE-ELANG1)500,445,445
445   CONTINUE
      IF(ANGLE - ELANG2)450,460,460
450   CONTINUE
      DTIME=DTIME*(ANGLE - ELANG1)/(ELANG2 - ELANG1)
460   CONTINUE
      RTIME = RTIME + DTIME
500   CONTINUE
C     SET FLAG TO INDICATE THAT WE ARE LOOKING FOR 2ND VORTX
C     SET VELOCITIES NEGATIVE THAT CORRESPOND TO POINTS THAT WERE USED
C     TO DEFINE 1ST VORTEX
      IF(KOUNT-2)530,530,525
525   CONTINUE
      MINPTS=VORTOL*KOUNT
530   CONTINUE
      NBA=NUMPTS-KOUNT
      IF(NBA-MINPTS)2280,540,540
540   CONTINUE
          DO 550 L=1,KOUNT
          J=INDEX(L)
          IVEL(J)=- IVEL(J)
550       CONTINUE
          GO TO 2000
C     TRY ANOTHER PT WITH A MAX VELOCITY
1000  CONTINUE
      NSPIKE=NSPIKE + 1
      IF(NUMPTS-2)2280,2280,1001
1001  CONTINUE
      IF(LNOISE-NSPIKE)2300,2300,1010
```

```
 1010   CONTINUE
        IVEL(MAX1)= -IVEL(MAX1)
        MAX1= MAX2
        MAX2= 0
        IF(MAX1)2000,2000,1050
 1050   CONTINUE
        CALL GETVEL(IVEL(MAX1),VELMAX)
        IF(VELMAX)2280,2280,1060
 1060   CONTINUE
        IF(NVORTX)2, 2, 10
 2000   CONTINUE
C       FIND NEW MAXIMA FOR VORTEX SEARCH
        MAX1=1
        MAX2=2
        L=3
        IF(IVEL(MAX1)-IVEL(MAX2)) 2050,2100,2100
 2050   CONTINUE
        MAX1=2
        MAX2=1
 2100   CONTINUE
             DO 2200 L=3,NUMPTS
        IF(IVEL(MAX2) - IVEL(L)) 2130,2200,2200
 2130   CONTINUE
        IF(IVEL(MAX1) - IVEL(L))2140, 2150,2150
 2140   CONTINUE
             MAX2=MAX1
             MAX1=L
        GO TO 2200
 2150        MAX2=L
 2200        CONTINUE
        CALL GETVEL(IVEL(MAX1),VELMAX)
        IF(VELMAX)2280,2280,2250
 2250   CONTINUE
         IF(NVORTX-1) 2248,2220,2220
 2220   CONTINUE
        DUM1=IX(MAX1)-LXCG
        DUM2=IY(MAX1)-LYCG
        DUM1=DUM1*DUM1+DUM2*DUM2
        RADMN=NOISE*IRADI
        IF(SQRT(DUM1)-RADMN)2230,2230,2248
 2230   CONTINUE
        IVEL(MAX1)=- IVEL(MAX1)
        GO TO 2000
 2248   CONTINUE
        IF(NVORTX)2,2,10
 2280   CONTINUE
        VELMAX=0
 2290   CONTINUE
        KOUNT=0
 2300   CONTINUE
        NFL=NVORTX + 1
        XCG(NFL)=0
        YCG(NFL)=0
        NOISES(NFL)=NSPIKE - NOISES(1)
        NCORPT(NFL)= KOUNT
        PKVEL(NFL)= VELMAX
```

```
2500   CONTINUE
 3100 IF (WTEST .NE. WHEEL) GO TO 3600
      IF(NVORTX)3600,3600,2700
2700   CONTINUE
 3110 IF  (IFRM) 3125,3600,3150
3125   IF(KVAN2-30)3130,3225,3600
3130   CONTINUE
       KVAN2=KVAN2 + 1
       TOTX2=TOTX2 + XCG1
       TOTY2=TOTY2+YCG1
       SSQX2=SSQX2+XCG1*XCG1
       SSQY2=SSQY2+YCG1*YCG1
       GO TO 3600
3150   IF(KVAN1-30)3155,3250,3600
3155   CONTINUE
       KVAN1=KVAN1 + 1
       TOTX1 = TOTX1 + XCG1
       TOTY1=TOTY1+YCG1
       SSQX1=SSQX1+XCG1*XCG1
       SSQY1=SSQY1+YCG1*YCG1
       GO TO 3600
 3225 AVXCG=TOTX2
      AVYCG=TOTY2
      STDX=SSQX2
      STDY=SSQY2
      KBUFOT(3)=517
      GO TO 3275
 3250 AVXCG=TOTX1
      AVYCG=TOTY1
      STDX=SSQX1
      STDY=SSQY1
      KBUFOT(3)=5
 3275 CALL SETAD (KBUFT6,MSTAT)
      KBUFOT(1)=NS3
      KBUFOT(4)=LDT-LLDT*KFRM
      LAVXCG=AVXCG/30.0
      LAVYCG=AVYCG/30.0
      KSTDX=SQRT((30.0*STDX-AVXCG*AVXCG)/870.0)
      KSTDY=SQRT((30.0*STDY-AVYCG*AVYCG)/870.0)
      CALL DECD (LAVXCG,5,MSTAT4)
      CALL DECD (LAVYCG,5,MSTAT7)
      CALL DECD (KSTDX,3,MSTA13)
      CALL DECD (KSTDY,3,MSTA15)
      CALL DISPIO (4,12,KBUFOT)
      GO TO 3700
 3600 CONTINUE
      CALL DISPLA
 3700 RETURN
      END
```

```
      SUBROUTINE RTV (N)
      COMMON /ATLP/ IBASE,DIMX
      COMMON/BUFFR/IBFR(12)
      COMMON /CNTRL/ KPROC, TACQTL, MTIRF
      COMMON /DSPL/ NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON /SETUP/ DATE,JDFT(20,6),JPLTP(3),IPLIND,IL1(60),TIM(2),
     1 IFLB(3),ILOV,IAHC,IL2(90),KRBUF(6)
      COMMON/IREAD/IREAD
      COMMON/IDUMMY/NB4,NFL
      COMMON /LOVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON /CFLI/ CFLI
      INTEGER CFLI
      DIMENSION N(6)
      DIMENSION NNN(4)
      EQUIVALENCE (IBFR4,IBFR(4)),(IBFR10,IBFR(10))
      EQUIVALENCE (IL01,IL2(1)),(IL3,IL2(3)),(IL6,IL2(6)),(IL8,IL2(8))
      EQUIVALENCE (IL31,IL2(31)),(IL33,IL2(33)),(IL36,IL2(36)),
     1 (IL38,IL2(38))
      EQUIVALENCE (IL42,IL2(42)),(IL44,IL2(44)),(IL47,IL2(47)),
     1 (IL49,IL2(49)),(IL52,IL2(52)),(IL54,IL2(54)),(IL57,IL2(57)),
     2 (IL59,IL2(59)),(IL62,IL2(62)),(IL64,IL2(64)),(IL67,IL2(67)),
     3 (IL69,IL2(69)),(IL72,IL2(72)),(IL74,IL2(74)),(IL77,IL2(77)),
     4 (IL79,IL2(79))
      DATA IND /0/
      DATA KM1/-1/
      CALL DISPIO(13,32)
      N2=N(2)
      IF (N2 .GE. 3) GO TO 400
      IL42=275
      IL44=275
      IL47=268
      IL49=268
      IL52=231
      IL54=231
      IL57=224
      IL59=224
      IL62=187
      IL64=187
      IL67=180
      IL69=180
      IL72=143
      IL74=143
      IL77=136
      IL79=136
      KPROC=1
      NS4=3
      IF (TACQTL .NE. 1.0E+6) GO TO 100
      IL01=910
      IL3=980
      IL6=910
      IL8=980
      GO TO 200
  100 CONTINUE
      IL01=IBASE
      IL3=IBASE+(TACQTL*DIMX/300.)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      IL6=IBASE
      IL8=IL3
  200 CONTINUE
      KT=3
      IF (MTIRF .EQ. 1) GO TO 300
      IL31=560
      IL33=700
      IL36=560
      IL38=700
      GO TO 500
  300 CONTINUE
      IL31=770
      IL33=910
      IL36=770
      IL38=910
      GO TO 500
  400 CONTINUE
      IBFR4=613
      IBFR10=571
      IF (IND .EQ. 1) GO TO 410
      IND=1
      CALL OPEN (1,1)
      NBA=1
      CALL SFUN(1,4,NBA,NFL)
  410 CONTINUE
      MTIRF=1
      KPROC=2
      IF(IREAD)425,420,425
420   CONTINUE
      IREAD=1
      CALL VREAD
      IPLIND=IPLN
      CFLI=1
      NNN(2)=IPLIND
      NNN(4)=0
      CALL PTYP (NNN)
      CFLI=0
425   CONTINUE
      IL01=400
      IL3=400
      IL6=400
      IL8=400
      IL31=400
      IL33=400
      IL36=400
      IL38=400
      IL42=319
      IL44=319
      IL47=312
      IL49=312
      IL52=275
      IL54=275
      IL57=268
      IL59=268
      IL62=231
      IL64=231
```

```
      IL67=224
      IL69=224
      IL72=187
      IL74=187
      IL77=180
      IL79=180
      KT=10
500   CONTINUE
      CALL DISPIO(13,64)
      CALL DISPIO (4,24,IBFR)
      IBFR4=657
      IBFR10=615
      CALL DISPIO (2,KT,KM1)
      CALL DISPIO (9,12,KRBUF)
      RETURN
      END
```

```
      SUBROUTINE DISPLA
C     IDITTB(1) = INITIAL X DIT POS FOR TABLE FOR VAN1 DATA
C     IDITTB(2) = INITIAL X DIT POS FOR TABLE FOR VAN2 DATA
C     IYDIT(1)  = INITIAL Y DIT POS FOR XY PLOT FOR VAN1 DATA
C     IYDIT(2)  = INITIAL Y DIT POS FOR XY PLOT FOR VAN2 DATA
C     IDITY(1)  = INITIAL Y DIT POS FOR Y-TIME PLOTS FOR VAN1 DATA
C     IDITY(2)  = INITIAL Y DIT POS FOR Y-TIME PLOTS FOR VAN2 DATA
C     IDITTX(1) = INITIAL TIME DIT POS FOR TIME-X PLOTS FROM VAN1 DATA
C     IDITTX(2) = INITIAL TIME DIT POS FOR TIME-X PLOTS FROM VAN2 DATA
C     IRUNXY = DIT POS FOR CENTER OF RUNWAY FOR XY PLOTS
C     IRUNXT = DIT POS FOR CENTER OF RUNWAY FOR XT PLOTS
C     IDITTM = DIT POS FOR INITIAL TIME POS FOR Y-TIME PLOTS
C     LOCVAN(1)=LOCATION OF VAN1 WRT CENTER OF RUNWAY
C     LOCVAN(2)=LOCATION OF VAN2 WRT CENTER OF RUNWAY
C     NS1  SCREEN FLAG FOR XY PLOTS
C     NS2  SCREEN FLAG FOR TIME PLOTS
C     NS3  SCREEN FLAG FOR TABLE
C     XRATXY = RATIO FOR X DIT POS FOR XY PLOTS
C     YRATXY = RATIO FOR Y DIT POS FOR XY PLOTS
C     XRATXT = RATIO FOR X DIT POS FOR    X-TIME PLOTS
C     TRATXT = RATIO FOR TIME DIT POS FOR X-TIME PLOTS
C     YRATYT = RATIO FOR Y    DIT POS FOR Y-TIME PLOTS
C     TRATYT = RATIO FOR TIME DIT POS FOR Y-TIME PLOTS
      COMMON/INITL/ ITPLT, PORTS(2), STARBS(2),VELMNF
      COMMON /DSPL/ NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON /LDVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON/ IADDRS/ IADDXY(26), IADPTS(2),IDATA(66), IADDAT,BLANK,
     1 IABLNK
       COMMON            XCG(2), YCG(2), INDEX(250), NCORPT(2), NOISES(2),
     1                    PKVEL(2),ELVANG(2),NVORTX, KFRM,RTIME
      COMMON /ROUT/ ROUT,KGBF
      DIMENSION IDITTB(2), IYDIT(2), IDITY(2), IDITTX(2),LOCVAN(2)
      DIMENSION IBBUFF(24), PORTS(2), STARBS(2)
      INTEGER BLANK
      BYTE IDATA, DECIML, KJJK,NEG
      DATA DECIML/ ' '/
      DATA NEG/'-'/
      DATA LOCVAN/  400, 400/
      DATA IBBUFF(2), IBBUFF(3)/1,0/
      DATA DSPM/'DATA'/
      DATA SNAME/ 'DISP'/
C     REMOVE AFTER TEST  *****
      DATA IRUNXY/ 518/
      DATA XRATXY/ 7/
      DATA YRATXY/ 88/
      DATA IYDIT/459,85/
      DATA IDITTB/ 0,512/
      DATA IDITY/459,63/
      DATA IDITTX/712,316/
      DATA IRUNXT/784/
      DATA IDITTM/70/
      DATA XRATXT/ 42/
      DATA TRATXT/3 1625/
      DATA YRATYT/ 88/
      DATA TRATYT/4 2/
```

```
C        REMOVE    ************************
      ROUT= SNAME
      ITPLT=KFRM
      IF (KFRM .LT. 26)GO TO 10
      ITPLT=ITPLT/26
      ITPLT=KFRM - 26 * ITPLT + 1
 10   CONTINUE
      MINANG= ELVANG(1)
      MAXANG=ELVANG(2)
C     ORDER ANGLES
      IF(MAXANG.GE.MINANG) GO TO 20
      IDUM=MINANG
      MINANG=MAXANG
      MAXANG=IDUM
20    CONTINUE
      IF(NVORTX.GT.0) GO TO 100
C     NO VORTEX FOUND
      IXC1=0
      IXC2=0
      IYC1=0
      IYC2=0
      LVEL=PKVEL(1)
      IRVEL=0
      IVAN=1
      IF(IFRM.LT.0)IVAN=2
      GO TO 1000
100   CONTINUE
C     TRANSFORM TO CENTER OF RUNWAY COORDINATE SYS.
      IF(IFRM.GT.0) GO TO 150
C     DATA REC FROM VAN2
      IVAN=2
      IXC1= LOCVAN(IVAN) - XCG(1)
      IXC2= LOCVAN(IVAN) - XCG(2)
      GO TO 175
150   CONTINUE
C     DATA REC FROM VAN1
      IVAN=1
      IXC1= XCG(1) - LOCVAN(IVAN)
      IXC2= XCG(2) - LOCVAN(IVAN)
175   IF(NVORTX.EQ.2) GO TO  200
      IF(IVAN.EQ.1) GO TO 180
C     ONLY ONE VORTEX WAS FOUND
      IF(( PORTS(IVAN) - XCG(1)).LE.(XCG(1)-STARBS(IVAN)))GO TO 300
      GO TO 210
 180  CONTINUE
      IF(( PORTS(IVAN) - XCG(1)).LE.(XCG(1)-STARBS(IVAN)))GO TO 210
      GO TO 300
C     DATA WILL BE ORDERED SO THAT LEFT VORTEX WILL BE IN LOC. 1
200   CONTINUE
      IF( IXC1.LE. IXC2) GO TO 300
C     VORTICES IN REV. POS.  SO SWITCH
210   CONTINUE
      IDUM=IXC1
      IXC1=IXC2
      IXC2=IDUM
      IYC1=YCG(2)
      IYC2=YCG(1)
```

```
      LVEL=PKVEL(2)
      IRVEL=PKVEL(1)
      IDUM=NOISES(1)
      NOISES(1)=NOISES(2)
      NOISES(2)=IDUM
      IDUM=NCORPT(1)
      NCORPT(1)=NCORPT(2)
      NCORPT(2)=IDUM
      DUM=XCG(1)
      XCG(1)=XCG(2)
      XCG(2)=DUM
      DUM=YCG(1)
      YCG(1)=YCG(2)
      YCG(2)=DUM
      GO TO 1000
300   CONTINUE
      IYC1=YCG(1)
      IYC2=YCG(2)
      LVEL=PKVEL(1)
      IRVEL=PKVEL(2)
C     CALCULATE DISTANCE FROM THE CENTER OF THE RUNWAY
1000  CONTINUE
C     DATA IS NOW IN CORRECT LOCATIONS
      IF(NS3.EQ.3) GO TO  1500
C     TABLE WAS CHOSEN AS A DISPLAY OPTION
      IDUM= 01*KFRM
      IDUM1=KFRM-100*IDUM
      IDUM= 1*IDUM1
      IDATA(1)=IDUM+48
      IDATA(2)=IDUM1-10*IDUM+48
      IDUM=  001*NUMPTS
      IDATA(4)=IDUM + 48
      IDUM1=NUMPTS - 1000*IDUM
      IDUM=  01*IDUM1
      IDATA(5)= IDUM + 48
      IDUM1= IDUM1 - 100*IDUM
      IDUM=  1*IDUM1
      IDATA(6)= IDUM+ 48
      IDATA(7)= IDUM1 - 10*IDUM + 48
      IDUM=  01*NCORPT(1)
      IDATA(9)= IDUM + 48
      IDUM1= NCORPT(1) - 100*IDUM
      IDUM=  1*IDUM1
      IDATA(10)=IDUM + 48
      IDATA(11)=IDUM1 - 10*IDUM + 48
      IDUM=  01*NCORPT(2)
      IDATA(13)=IDUM + 48
      IDUM1= NCORPT(2) - 100*IDUM
      IDUM=  1*IDUM1
      IDATA(14)=IDUM + 48
      IDATA(15)=IDUM1 - 10*IDUM + 48
      IDUM=  1*NOISES(1)
      IDATA(18)=IDUM + 48
      IDATA(19)= NOISES(1) - 10*IDUM + 48
      IDUM=  1*NOISES(2)
      IDATA(21)=IDUM + 48
      IDATA(22)= NOISES(2) - 10*IDUM + 48
```

```
      IDUM= .1*MINANG
      IDATA(25)= IDUM + 48
      IDATA(26)= MINANG - 10*IDUM + 48
      IDUM= .1*MAXANG
      IDATA(28)=IDUM + 48
      IDATA(29)= MAXANG - 10*IDUM + 48
      IDUM= .01*LVEL
      IDATA(32)=IDUM + 48
      IDUM1= LVEL - 100*IDUM
      IDUM= .1*IDUM1
      IDATA(33)=IDUM + 48
      IDATA(34)= IDUM1 - 10*IDUM + 48
      IDUM= .01*IRVEL
      IDATA(36)=IDUM + 48
      IDUM1=IRVEL - 100*IDUM
      IDUM= .1*IDUM1
      IDATA(37)=IDUM + 48
      IDATA(38)= IDUM1 - 10*IDUM + 48
       KTIME=RTIME
      IDUM= .01*KTIME
      IDATA(40)= IDUM + 48
      IDUM1=KTIME-100*IDUM
      IDUM= .1*IDUM1
      IDATA(41)=IDUM + 48
      IDATA(42)=IDUM1 - 10*IDUM + 48
      IDATA(43)=DECIML
      IDUM=RTIME
      IDUM=  RTIME*10.0 - 10*IDUM
      IDATA(44)=IDUM + 48
      IF (IXC1.GE.0) GO TO 1200
      KDUM= IABS(IXC1)
      IDATA(46)=NEG
      GO TO 1250
1200  CONTINUE
      KDUM=IXC1
      IDATA(46)=BLANK
1250  CONTINUE
      IDUM= .01*KDUM
      IDATA(47)= IDUM + 48
      IDUM1= KDUM - 100*IDUM
      IDUM= .1*IDUM1
      IDATA(48)= IDUM + 48
      IDATA(49)= IDUM1 - 10*IDUM + 48
      IDUM= .01*IYC1
      IDATA(51)= IDUM + 48
      IDUM1= IYC1 - 100*IDUM
      IDUM= .1*IDUM1
      IDATA(52)= IDUM + 48
      IDATA(53)= IDUM1 - 10*IDUM + 48
      IF (IXC2.GE.0) GO TO 1260
      KDUM=IABS(IXC2)
      IDATA(56)=NEG
      GO TO 1270
1260  CONTINUE
      KDUM=IXC2
      IDATA(56)=BLANK
1270  CONTINUE
      IDUM= .01*KDUM
```

```
      IDATA(57)= IDUM + 48
      IDUM1= KDUM - 100*IDUM
      IDUM= 1*IDUM1
      IDATA(58)= IDUM + 48
      IDATA(59)= IDUM1 - 10*IDUM + 48
      IDUM=  01*IYC2
      IDATA(61)= IDUM + 48
      IDUM1 = IYC2 - 100*IDUM
      IDUM= 1*IDUM1
      IDATA(62)= IDUM + 48
      IDATA(63)= IDUM1 - 10*IDUM + 48
      IF(NVORTX EQ 2) GO TO 1450
      IF(NVORTX EQ 1) GO TO 1300
      IDATA(13)=BLANK
      IDATA(14)=BLANK
      IDATA(15)=BLANK
      IDATA(21)=BLANK
      IDATA(22)=BLANK
      IDATA(36)=BLANK
      IDATA(37)=BLANK
      IDATA(38)=BLANK
      DO 1280 J=40,49
      IDATA(J)=BLANK
1280  CONTINUE
      DO 1285 J=51,59
      IDATA(J)=BLANK
1285  CONTINUE
      IDATA(61)=BLANK
      IDATA(62)=BLANK
      IDATA(63)=BLANK
      GO TO 1450
1300  CONTINUE
      IF(XCG(1) EQ 0) GO TO 1350
      IDATA(56)=BLANK
      IDATA(57)=BLANK
      IDATA(58)=BLANK
      IDATA(59)=BLANK
      IDATA(61)=BLANK
      IDATA(62)=BLANK
      IDATA(63)=BLANK
      GO TO 1450
1350  CONTINUE
      IDATA(46)=BLANK
      IDATA(47)=BLANK
      IDATA(48)=BLANK
      IDATA(49)=BLANK
      IDATA(51)=BLANK
      IDATA(52)=BLANK
      IDATA(53)=BLANK
      GO TO 1450
1450  CONTINUE
      IBBUFF(1)=N33
      IBBUFF(3)=IDITTB(IVAN)
      IBBUFF(4)=LDT-LLDT*KFRM
      IF(KFRM. GT  55)IBBUFF(4)=LDT-LLDT*(KFRM-55)
      IF(KFRM. GT. 110)IBBUFF(4)=LDT-LLDT*(KFRM-55)
```

```
      IBBUFF(5)=63
      IBBUFF(6)=IADDAT
      CALL DISPIO(4,12,IBBUFF)
 1500 CONTINUE
      IF(NVORTX.EQ.0) GO TO 4000
      IF(NS1.EQ.3) GO TO 2000
C     XY PLOTS WERE CHOSEN AS A DISPLAY OPTION
      IBBUFF(1)=NS1
      IBBUFF(5)=1
      IBBUFF(7)=NS1
      IBBUFF(8)=IBBUFF(2)
      IBBUFF(11)=1
      IF(XCG(1).EQ.0.0) GO TO 1700
C     THERE IS A PORT VORTEX TO BE PLOTTED
      IBBUFF(3)= IRUNXY + IXC1*XRATXY
      IBBUFF(4)= IYDIT(IVAN) + YCG(1)*YRATXY
      IBBUFF(6)= IADDXY(ITPLT)
      GO TO  1800
 1700 CONTINUE
C     THERE IS NO PORT VORTEX
      IBBUFF(6)=IABLNK
 1800 CONTINUE
      IF(XCG(2).EQ.0.0) GO TO 1900
C     THERE IS A STARBOARD VORTEX
      IBBUFF(9)=IRUNXY + IXC2*XRATXY
      IBBUFF(10)=IYDIT(IVAN) + YCG(2)*YRATXY
      IBBUFF(12)=IADDXY(ITPLT)
      GO TO 1950
 1900 CONTINUE
C     THERE IS NO STARBOARD VORTEX
      IBBUFF(12)=IABLNK
 1950 CONTINUE
      CALL DISPIO(4,24,IBBUFF)
 2000 CONTINUE
      IF(NS2.EQ.3) GO TO 4000
C     X-TIME AND Y-TIME PLOTS WERE CHOSEN AS A DISPLAY OPTION
      IF(RTIME.GE.81.) GO TO 4000
      IBBUFF(1)= NS2
      IBBUFF(7)= NS2
      IBBUFF(13)=NS2
      IBBUFF(19)=NS2
      IBBUFF(5) = 1
      IBBUFF(11)= 1
      IBBUFF(17)= 1
      IBBUFF(23)= 1
      IBBUFF(4) = IDITTX(IVAN) - RTIME*TRATXT
      IBBUFF(16)= IBBUFF(4)
      IBBUFF(9) = IDITTM +        RTIME*TRATYT
      IBBUFF(21)= IBBUFF(9)
      IF(XCG(1).EQ.0.0) GO TO 2500
C     THERE IS A PORT VORTEX
      IBBUFF(3)= IRUNXT +  IXC1*XRATXT
      IBBUFF(6)= IADPTS(1)
      IBBUFF(10)=IDITY(IVAN) + YCG(1)*YRATYT
      IBBUFF(12)=IBBUFF(6)
      GO TO 3000
```

```
2500  CONTINUE
C     THERE WAS NO PORT VORTEX
      IBBUFF(6)=IABLNK
      IBBUFF(12)=IABLNK
3000  CONTINUE
      IBBUFF(14)=IBBUFF(2)
      IBBUFF(8) =IBBUFF(2)
      IBBUFF(20)=IBBUFF(2)
      IF(XCG(2).EQ.0.0) GO TO 3500
C     THERE IS A STARBOARD VORTEX
      IBBUFF(15)= IRUNXT + IXC2*XRATXT
      IBBUFF(18)= IADPTS(2)
      IBBUFF(22)= IDITY(IVAN) + YCG(2)*YRATYT
      IBBUFF(24)= IADPTS(2)
      IF(XCG(1).NE.0.)GO TO 3600
      IBBUFF(18)=IADDXY(19)
      IBBUFF(24)=IADDXY(19)
      GO TO 3600
3500  CONTINUE
C     THERE IS NO STARBOARD VORTEX
      IBBUFF(6)=IADDXY(19)
      IBBUFF(12)=IADDXY(19)
      IBBUFF(18)= IABLNK
      IBBUFF(24)= IABLNK
3600  CONTINUE
      CALL DISPIO(4,48,IBBUFF)
4000  CONTINUE
      IF(NVORTX.LT.2)RETURN
      PORTS(IVAN)=XCG(1)
      STARBS(IVAN)=XCG(2)
      RETURN
      END
```

```
      SUBROUTINE GETVEL(IV,VEL)
      IF(IV.LE.0) GO TO 1000
      CALL SUBBIT(IV,2,7,IDUM)
      IF(IDUM.GT.69) GO TO 100
      VEL= FLOAT(IDUM)*1.8
      RETURN
100   CONTINUE
      IF(IDUM.GT.75) GO TO 200
      VEL= 1.8*69.0 + FLOAT(IDUM - 69)*3.6
      RETURN
200   VEL = 1.8*69.0 + 3.6*6.0 + FLOAT(IDUM-75)*7.2
      RETURN
1000  VEL=0.0
      RETURN
      END
```

```
      SUBROUTINE DFLT (N)
      COMMON /BUFFER/ IBUF1(12),IBUF2(12),IBUFFF(2)
      COMMON /DSPL/ NS1,NS2,NS3,NS4,LDT,KSM,KLG,LLDT
      COMMON /IFLDV/ IFLDV
      COMMON/INPT/  PTTOL,NOISE,  VELTOL,  VORTOL,  IRADI,IRADI2,NRADI,
     1   CONXY,STIME,FRVTOL
      COMMON /LDVDAT/IFLY,IFRM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON/OWT/IOWT(6)
      COMMON /ROUT/ ROUT,KGBF
      COMMON /SETUP/ DATE,JDFT(20,6),JPLTP(3),IPLIND,IL1(60),TIM(2),
     1 IFLB(3),ILDV,IAHC,IL2(90),KRBUF(6)
      COMMON/TMPS/KDFTT(6),JAUTO
      BYTE IBUF2
      DIMENSION N(8)
      EQUIVALENCE (IBUF01,IBUF2(1)),(IBUF3,IBUF2(3)),(IBUF5,IBUF2(5)),
     1 (IBUF7,IBUF2(7)),  (IBUF9,IBUF2(9)),(IBUF11,IBUF2(11))
      EQUIVALENCE (KDFTT1,KDFTT(1)),(KDFTT2,KDFTT(2)),(KDFTT3,KDFTT(3)),
     1 (KDFTT4,KDFTT(4)),(KDFTT5,KDFTT(5)),(KDFTT6,KDFTT(6))
      DATA DATS/'DFLT'/
      ROUT = DATS
      N2=N(2)
      IF ((N2 .GE. 10) .OR. (N2 .EQ. 7)) GO TO 900
      IF (N2 .GT. 6) GO TO 500
      KDFTT(N2)=((N(5)-420.)*100.0)/560.0
      IL1(10*N2-7)=N(5)
      IL1(10*N2-2)=N(5)
      GO TO (25,50,75,100,125,150),N2
   25 CONTINUE
      IRADI =KDFTT1/CONXY
      GO TO 900
   50 VELTOL = KDFTT2*0.01
      GO TO 900
   75 PTTOL = KDFTT3*0.01
      GO TO 900
  100 NOISE = KDFTT4/10
      GO TO 900
  125 VORTOL = KDFTT5*0.01
      GO TO 900
  150 FRVTOL = KDFTT6*0.01
      GO TO 900
500   DO 525 I=1,6
      JDFT(IPLIND,I)=KDFTT(I)
  525 CONTINUE
      GO TO 950
  900 CONTINUE
      KI=IRADI*CONXY
      CALL DECD (KI,2,IBUF01)
      KI=VELTOL*100.0
      CALL DECD (KI,2,IBUF3)
      KI=PTTOL*100.0
      CALL DECD (KI,2,IBUF5)
      CALL DECD (NOISE,2,IBUF7)
      KI=VORTOL*100.0
      CALL DECD (KI,2,IBUF9)
      KI=FRVTOL*100.0
      CALL DECD (KI,2,IBUF11)
      CALL DISPIO(9,12,IOWT)
  950 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RTYP(N)
      COMMON /BUFFER/ IBUF1(12),IBUF2(6),IBUF3(2)
      COMMON /CELI/ CELI
      COMMON /CONTRL/ KPROC, IACCTL, MTIRF
      COMMON /DELTPT/KDEFLT
      COMMON /IELDV/ IELDV
      COMMON/INPT/  PTTOL,NOISE, VELTOL, VORTOL, IRADI,IRADI2,NRADI,
     1  CONXY,STIME,FRVTOL
      COMMON /LDVDAT/IELY,IERM,ITMINT(2),ITMEND(2),IDAY,IPLN,NUMPTS,
     1 IX(1000),IY(1000),INTENS(1000),IVEL(1000)
      COMMON /OWT/ IOWT(6)
      COMMON /PLNAME/ IDAT(60)
      COMMON /ROUT/ ROUT,KGRE
      COMMON /SETUP/ DATE,JDET(20,6),JPLTP(3),IPLIND,IL1(60),TIM(2),
     1 IELB(3),ILDV,IAHC,IL2(90),KRBUF(6)
      COMMON /TMPS/  KDETT(6),  JAUTO
      INTEGER CELI
      INTEGER Y1,YM,YT,DY,DDY
      DIMENSION M(2),N(6)
      DIMENSION IDATE(2)
      EQUIVALENCE (IDATE(1),DATE)
      EQUIVALENCE (JPLTP1,JPLTP(1)),(JPLTP2,JPLTP(2)),(JPLTP3,JPLTP(3))
C
C        Y1 IS LOW Y-COORDINATE
C        YM IS HIGH Y-COORDINATE
      DATA Y1,YM/420,980/
C
      DATA KM1/-1/
      DATA DATS/'RTYP'/
      ROUT = DATS
      N2=N(2)
      IF (N2 .EQ. 22) GO TO 100
      IF (N2 .NE. 21) GO TO 75
      IF (KPROC.EQ.1) GO TO 995
      KDEFLT=1
      GO TO 930
 75   CONTINUE
      IF (N(4) .EQ. 1) GO TO 200
      IPLIND=N2
      I=3*IPLIND-2
      JPLTP1=IDAT(I)
      JPLTP2=IDAT(I+1)
      JPLTP3=IDAT(I+2)
      GO TO 900
100   CONTINUE
      I=2
      M(2)=1
      IF (KPROC .EQ. 1) GO TO 110
      I=9
      M(2)=3
 110  CONTINUE
      CALL DISPIO (2,I,0)
      CALL RTV (M)
      RETURN
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
  200 CONTINUE
      IPLIND=18
      JPLTP1=N(5)
      JPLTP2=N(6)
      JPLTP3=N(7)
  900 CONTINUE
      IF(KPROC-1) 930,930,910
910   CONTINUE
912   IF(IPLIND - IPLN) 915, 930,915
915   CONTINUE
      NBA=1
      CALL SFUN(1,10,NBA,NFL)
      CALL VREAD
      GO TO 912
  930 CONTINUE
      DX = XM-X1
      DO 950 I=1,6
      KDFTT(I) = JDFT(IPLIND,I)
      KX=X1+(DX*KDFTT(I))*0.01
      IL1(10*I-7)=KX
      IL1(10*I-2)=KX
  950 CONTINUE
      IRADI = KDFTT(1)/CONXY
      VELTOL = KDFTT(2)*0.01
      PTTOL = KDFTT(3)*0.01
      NOISE = KDFTT(4)/10
      VORTOL = KDFTT(5)*0.01
      FRVTOL = KDFTT(6)*0.01
      J=5
      IF (KPROC .EQ. 2) J=15
      DO 960 I=1,3
      IBUF1(I)=IFLB(I)
      IBUF1(I+3)=IDATE(I)
  960 IBUF1(I+9)=JPLTP(I)
      CALL TIME (IBUF1(6))
      IF (CFLI .EQ. 1) GO TO 995
      CALL DISPIO(2,J,0)
      CALL DISPIO(1,24,IBUF1)
C
C        CALL DISPLAY CONTROLLER TO DISPLAY DEFAULT IN WRITE THROUGH
      CALL DISPIO (2,J+1,KM1)
      CALL DISPIO (9,12,IOWT)
  995 CONTINUE
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY